Dossier de projet

2013

Pondeuse d'appels

En collaboration avec l'entreprise :

Mutuelles du Mans Assurances

Prieur Quentin Plateau Mathias Chrétien Clément Dauvergne Florian

Sommaire

1. Analyse technique 5
1.1 Le principe de la VOIP (Voice Over IP).....5
1.2 Le transport de la voix.....5
1.2.1 Les différents protocoles.....6
1.3 Mise en place de la maquette MMA.....7
1.3.1 Matériels à notre disposition 7
1.3.2 Présentation de la central d'appels.....7
2. Conception préliminaire : Partie Prieur Quentin (E4).....8
2.1 Situation dans le projet.....8
2.2 Le serveur Trixbox.....10
2.2.1 Qu'est-ce que trixbox.....10
2.2.2 Pourquoi ce choix 10
2.3 Les différents services du serveur Trixbox.....11
2.3.1 Le service DHCP.....11
2.3.1.a Qu'est-ce qu'un DHCP.....11
2.3.1.b Son fonctionnement 11
2.3.1.c Mise en place sur le serveur 12
2.3.2 Administration web : Création de comptes SIP.....13
2.3.3 Administration web : Enregistrement d'un guide.....13
2.3.4 Administration web : Installation d'une musique d'attente 14
2.3.5 Administration web : Installation d'une file d'attente.....15
2.3.6 Administration web : Installation d'un IVR 18
2.3.6.a Qu'es qu'un IVR?.....18
2.3.6.b Configuration.....18
2.3.7 Le protocole FTP.....19
2.3.7.a Qu'est-ce qu'un FTP.....19
2.3.7.b Son Fonctionnement 19
2.3.8 PhpMyAdmin.....20
2.3.8.a Qu'est-ce que phpMyAdmin ? 20
2.4 Étude d'un Trunck SIP.....21
2.4.1 Qu'est-ce qu'un Trunk SIP ?.....21
2.5 Mise en place de la maquette MMA: Partie technique 21
2.5.1 Visuelle de la maquette.....22
2.5.2 Première installation 23
2.5.3 Choix du protocole 24
3. Conception préliminaire : Partie Plateau Mathias (E1).....26
3.1 L'architecture 26
3.2 Architecture logicielle :.....28
3.2.1 Choix de la base de données 28
3.2.2 Les couches logicielles 28
3.2.3 Choix du protocole 28
3.3 Mon rôle au sein du projet 30
3.4 Les interfaces homme/ machine 33
3.5 Les différents diagrammes : 36
3.5.1 Le diagramme de classe.....43
3.6 Les interactions :.....45
3.7 Les tables de la base de données.....46
3.8 Travail futur :.....47

3.9 Diagramme de gantt....48
3.10 Conclusion :....49
4. Conception préliminaire : Partie Chrétien Clément (E2)....50
4.1 Étude préliminaire:Suivre les Appels simultanés....52
4.1.1 Diagramme d'état transition d'un essai....54
4.1.2 Les données de suivi....54
4.1.3 Diagramme de séquence pour un appel :....55
4.1.4 Les principales classes de l'application....55
4.2 Étude préliminaire:Générer un appel....58
4.2.1 Diagramme d'état transition d'un appel....60
4.2.2 Solution trouvée pour la génération d'appel à partir du serveur Asterisk....62
4.2.3 Exemple de fichier :....63
4.3 Échange de données 66
4.3.1 Étude des solutions possibles....66
4.4 Choix concernant le développement....66
4.4.1 Choix liés aux développements de l'application....66
4.4.1.a Langage utilisé....66
4.4.1.b Bibliothèque utilisée....66
4.4.1.c Choix de l'EDI....67
4.4.1.d Système d'exploitation....67
5. Conception préliminaire : Partie Dauvergne Florian (E3)....69
5.1 Qualité de service :....69
5.2 Remise en contexte....69
5.3 Qu'est-ce que la qualité de service ?....69
5.4 Paramètres QoS....69
5.5 Les logiciels d'analyse de la QoS....73
5.6 La librairie Pcap....75
5.7 Affichage de la QoS....76
6. Conception détaillée : Partie Prieur Quentin (E4) 78
6.1 Installation du serveur trixbox....79
6.2 Serveur FTP....80
6.2.1 Installation et configuration 80
6.2.1.a Ajout des utilisateurs au système....82
6.2.1.b Ajout des accès de connexion au FTP....82
6.2.1.c Autoriser les accès FTP à un compte virtuel : asterisk....83
6.3 phpMyAdmin....84
6.3.1.a Installation....84
6.3.1.b Connexion à la base MYSQL....85
6.4 Trunk SIP : Mise en place....86
6.4.1 Solution N°1....86
6.4.1.a Configuration : côté serveur Trixbox 86
6.4.1.b Configuration : côté OXE Alcatel....88
6.4.1.c Conclusion : 89
6.4.2 Solution N°2....90
6.4.2.a Configuration: Mise en place du trunk....90
6.4.2.b Configuration: redirection d'appels....91
6.4.2.c Conclusion sur la solution N°1....91
6.5 Conclusion....92
7. Conception détaillée : Partie Plateau Mathias (E1)....93
7.1 Configuration Qt/MySQL 93

7.2 Principe d'accès à la base de données avec Qt4 94
7.2.1 La connexion à la base de données:.....94
7.2.2 L'envoi d'une requête Sql préparée 95
7.2.3 Gestion des erreurs d'envoi de la requête :.....96
7.3 Fonctionnement de l'Interface Homme-Machine 96
7.4 Nouvelle campagne, la configuration :.....96
7.4.1 Paramétrage de la campagne :.....102
7.4.2 Paramétrage d'un scénario :.....104
7.4.3 Supprimer une campagne :.....115
7.5 Conclusion :.....118
8. Conception détaillée : Partie Chrétien Clément (E2).....119
8.1 Diagramme de Séquence pour la génération d'un appel :.....120
8.2 Diagramme de Classes.....122
8.3 Générer un appel.....126
8.3.1 La génération de fichier d'appel.....127
8.3.2 La connexion à la trixbox en FTP.....130
8.3.3 Application dans le programme principal de la pondeuse d'appels.....133
8.3.4 La génération d'un scénario.....133
8.3.5 recupererFichierExtension().....134
8.3.6 genererFichierScenario() :.....135
8.4 Conclusion :.....139
9. Conception détaillée : Partie Dauvergne Florian (E3).....140
9.1 Configuration analyse.....140
9.2 Pré-Requis.....141
9.3 Lancement analyse.....142
9.4 Analyse de la qualité de service.....143
9.4.1 Fonctionnement et mise en œuvre.....143
9.4.2 Capture.....151
9.4.3 Capture.....152
9.4.4 Filtrage UDP.....153
9.4.5 Filtrage SIP.....154
9.4.6 Filtrage RTP & RTCP.....155
9.5 Affichage de statistiques.....156
9.6 Enregistrement des statistiques.....157
9.7 Conclusion.....158
10. Fiches de test : Prieur Quentin.....159
10.1 Plan de test.....159
10.2 Procès verbal de test.....162
10.3 Plan de test structurel.....164
11. Fiche de test : Plateau Mathias.....166
12. Fiche de test : Chrétien Clément.....173
12.1.1 Test n°1 :.....178
12.1.2 Test n°2 :.....179
13. Fiche de test : Dauvergne Florian.....180
14. Manuel d'utilisation OptionServeur.....181
14.1 Principe de fonctionnement :.....181
14.2 Présentation :.....181
14.3 Modifications d'une ou plusieurs données :.....181
15. Manuel d'utilisation de l'interface homme-machine :.....182
15.1 Nouvelle campagne :.....182

15.2 Charger une ancienne campagne :.....184
15.3 Supprimer une campagne:.....185
16. Annexe 1 - Installation.....187
17. Annexe 2 – Procédure de suppression d'un utilisateur 189

1. Analyse technique

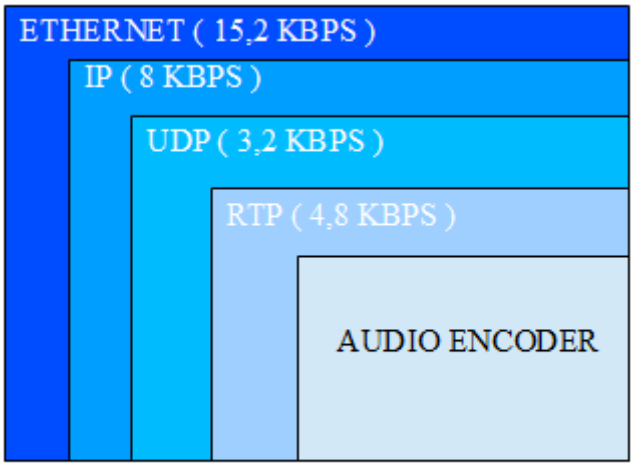
Dans le cadre de notre projet, qui a pour but de réaliser une pondeuse d'appels, nous devons nous renseigner sur les principes de la téléphonie classique et sur IP.

1.1 Le principe de la VOIP (Voice Over IP)

La voix sur IP, ou « VoIP », est une technique qui permet de communiquer par la voix (ou via des flux multimédias: audio et/ou vidéo) sur des réseaux compatibles IP, qu'il s'agisse de réseaux privés ou d'Internet, filaire (câble/ADSL/optique) ou non (satellite, WiFi, GSM, UMTS). Cette technologie est notamment utilisée pour prendre en charge le service de téléphonie sur IP (« ToIP » pour Telephony over Internet Protocol).

1.2 Le transport de la voix

Le transport de la voix sur IP est relativement complexe. La première étape est la numérisation du signal acoustique. C'est le codage. Ensuite, les informations sont découpées en trames pouvant circuler sur un réseau informatique. Divers protocoles peuvent alors être utilisés pour acheminer les informations au(x) destinataire(s). Ainsi le protocole RTCP[1] est utilisé pour contrôler le transport des paquets RTP[2].



Au total 31,2 KBPS

Le transport de communication sur IP est très dépendant du délai de latence d'un réseau. Ce délai influe beaucoup sur la qualité psychoacoustique d'une conversation. Avec l'avènement des réseaux 100 Mbit/s et ADSL, les temps de latence deviennent acceptables pour une utilisation quotidienne de la voix sur IP.

1 RTCP (Real-time Transport Control Protocol) repose sur des transmissions périodiques de paquets.
2 RTP (Real-Time Transport Protocol) est un protocole de communication informatique permettant le transport de données1 soumises à des contraintes de temps réel

1.2.1 Les différents protocoles

MGCP

Le protocole MGCP (Media Gateway Control Protocol) sert à l'échange de message de signalisation entre un contrôleur de passerelles de médias et des passerelles réparties dans un réseau IP. Pour l'établissement et la libération des connexions, MGCP se sert de signaux et d'événements. La standardisation de MGCP a été stoppée pour faire place à MEGACO/H.248 (MEdia GAteway COntrol protocol), protocole élaboré en collaboration entre l'IETF et l'UIT (RFC 2705). Ce nouveau standard n'étant pas dérivé de MGCP, la migration vers MEGACO/H.248 semble très difficile.

H323

La norme H.323, développée par l'IUT-T, est utilisée pour l'interactivité en temps réel (échange audio, vidéo, donné, contrôle et signalisation). C'est la norme la plus utilisée concernant la VoIP. Elle hérite de la norme H320 utilisée pour la voix sur RNIS. Comme toute norme, elle est constituée d'un ensemble de protocoles réalisant les différentes fonctions nécessaires à la communication.

SIP

Contrairement à la norme H323, SIP (Session Initiation Protocol) est un protocole unique de type requête/réponse très proche des protocoles HTTP et SMTP. Il commence à prendre le pas sur la norme H323. SIP est normalisé par l'IETF (RFC 3261). Il permet de créer et gérer des sessions entre participants pour échanger des données indépendamment de leur nature et du protocole de transport.

PABX

Commutateur téléphonique qui, à l'intérieur d'une entreprise, gère de manière automatique les communications entre plusieurs postes et qui sert à établir celles avec l'extérieur.

IPBX

Fonctionne sur le principe du PABX, utilise en plus le protocole internet (IP), en interne sur le réseau local (LAN) ou le réseau étendu (WAN) de l'entreprise.

1.3 Mise en place de la maquette MMA

Pour effectuer nos différents testent, Monsieur Lê, responsable communication au MMA est venue installer une maquette du centre d'appel actuellement en place dans les différents centres et bureaux.

1.3.1 Matériels à notre disposition

Pour réaliser ceci, l'entreprise nous a prêté différents périphériques :

- Quatre téléphones numériques de marque Alcatel-Lucent et de modèle UA 4035
- Un téléphone numérique de marque Alcatel-Lucent et de modèle 4039
- Un téléphone IP de marque Alcatel-Lucent, IPTouch 4038
- Un téléphone de type analogique
- Un PABX Alcatel-Lucent Omni PCX Enterprise R8.0
- Une carte CS
- Une carte GD-2

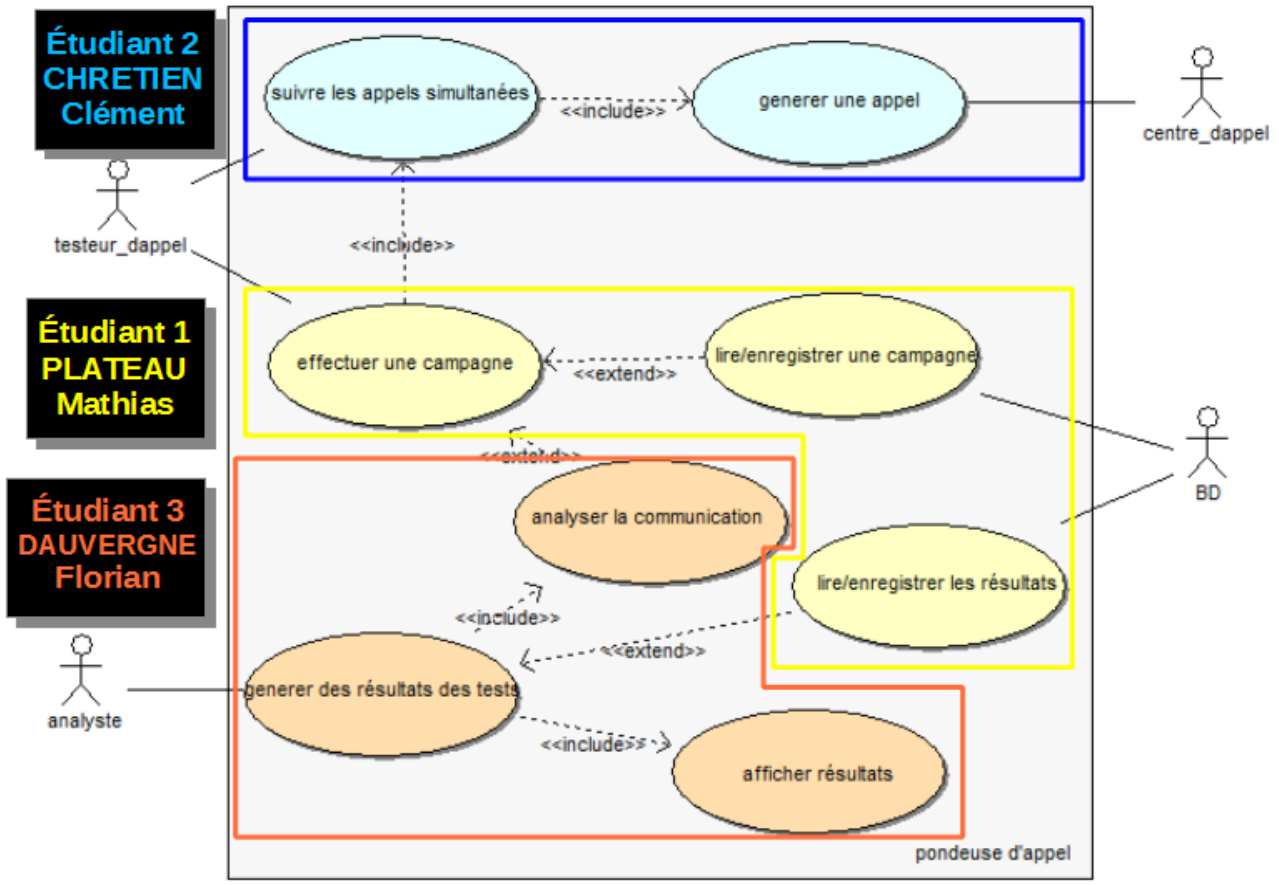
1.3.2 Présentation de la central d'appels

La centrale d'appels prêtée par les MMA, est composée d'un média gateway 9 positions, qui peut accueillir 9 cartes différentes. La carte GD (gateway driver) contrôle toutes les autres cartes, par exemple la CS (Call Server). Le système d'exploitation présent sur la CS est basé sur Mandriva Linux. Avec une sur couche propriétaire à Alcatel-Lucent. (CF. Annexe 1 - Installation)

2. Conception préliminaire : Partie Prieur Quentin (E4)

2.1 Situation dans le projet

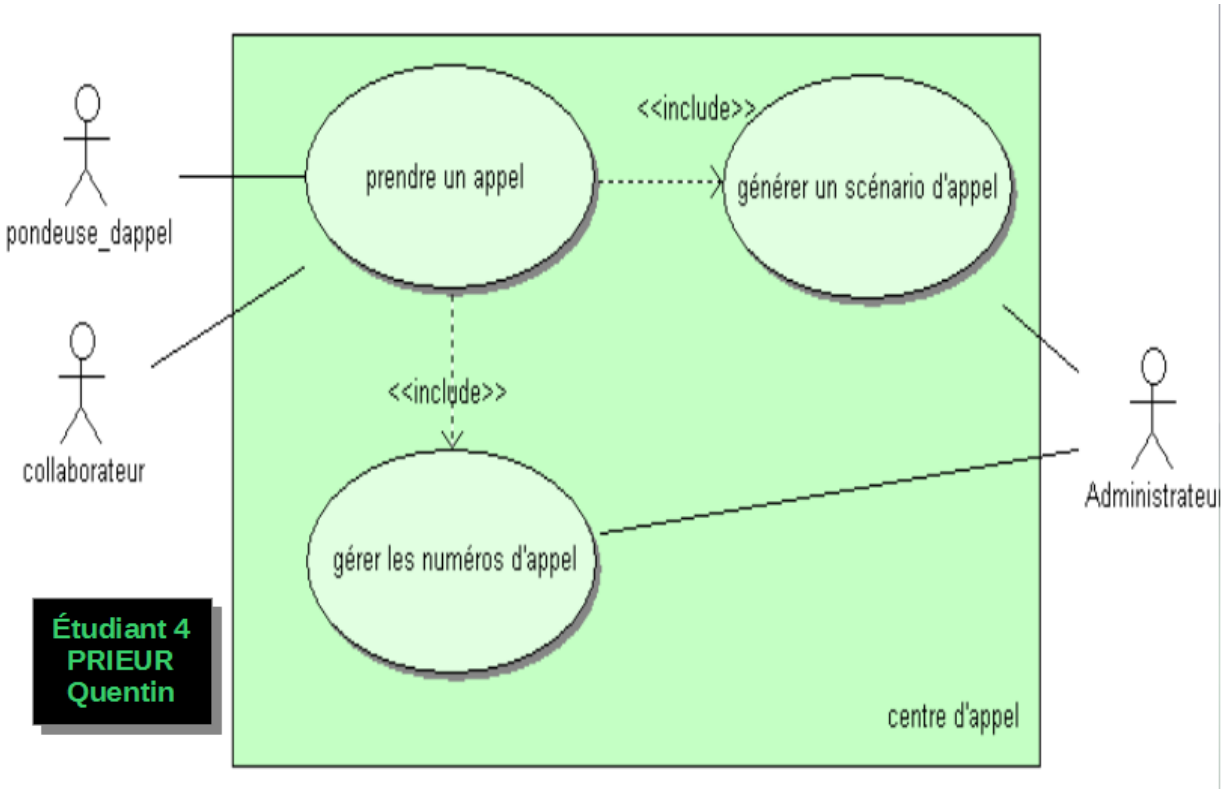
Dans le projet pondeuse d'appels, les tâches sont réparties entre 4 personnes :



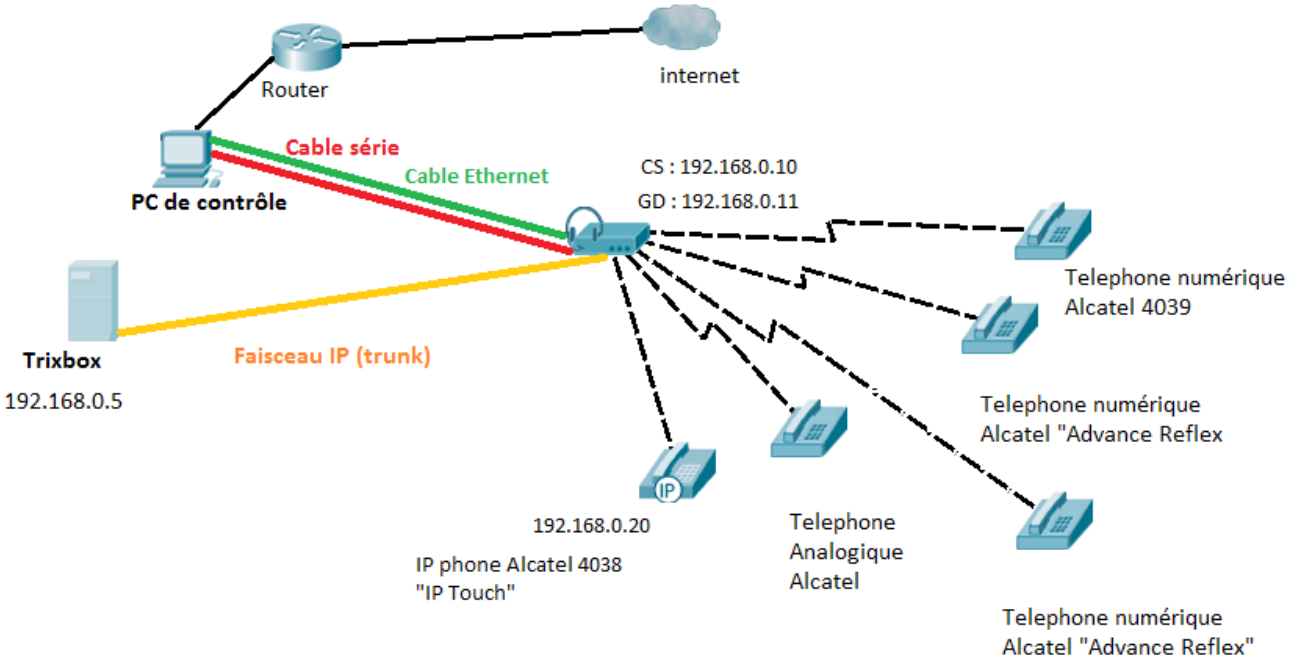
Chrétien Clément, représenté par l'étudiant 2 en **bleu** est en charge de la génération simultanée des appels, de la récupération des états de ceux-ci et du stockage de ces informations.

Plateau Mathias, repéré par l'étudiant 1 en **jaune,** est responsable de la création des Interfaces Homme Machine pour la génération des campagnes d'appels, ainsi que la sauvegarde de ces données.

Dauvergne Florian, indiqué par l'étudiant 3 en **orange,** s'est vu confier les tâches d'analyses, et de statistiques.



Pour ma part, je suis l'étudiant 4, en **vert** sur le diagramme. Je suis en charge de la partie centre d'appels. Je dois être capable de prendre un appel, et d'en sauvegarder les données du scénario joué par l'appelant.



Le centre d'appels, est composé de la CS et la GD, des téléphones qui y sont connectés,

et d'un pc de contrôle relié par deux connections: une carte Ethernet et un câble série. L'échange avec la pondeuse sera réalisé par un faisceau IP.

2.2 Le serveur Trixbox

2.2.1 Qu'est-ce que trixbox

Trixbox (connu auparavant sous le nom d'*Asterisk@Home*) est un logiciel libre d'autocommutateur téléphonique privé (PBX) ou IPBX basé sur le logiciel libre *Asterisk*.

En octobre 2006, le produit a été renommé T*rixbox* après que Digium, l'éditeur du produit *Astérisk*, eut demandé que le mot « asterisk » ne soit pas utilisé dans le nom du produit. Le changement de nom était d'autant plus justifié que le produit avait à cette époque beaucoup plus de fonctionnalité qu'*Astérisk*.

Trixbox CE est le logiciel qui a été téléchargé le plus souvent dans la liste des projets réalisés à partir du logiciel libre Asterisk (selon Sourceforge.net)

Trixbox CE est 100 % libre et sous licence GPLv2. Les membres fondateurs du projet *Trixbox CE* sont Kerry Garrison et Andrew Gillis.

Il existe également une version PRO de *trixbox*, décliné sous plusieurs versions :

- *Standard Edition (SE)* ;
- *Enterprise Edition (EE)* ;
- *Call Center Edition (CCE)* ;

Le CD Trixbox inclut le noyau CentOS pour le système d'exploitation, Asterisk, pour la partie IPBX et interface web, et Flash Operator Panel (FOP) pour la partie graphique de l'interface web.

Une fois le produit installé, l'administration de Trixbox est entièrement réalisée depuis une interface web. Seul un accès SSH peut être parfois utile lors de l'ajout de nouveaux modules fonctionnels,

2.2.2 Pourquoi ce choix

Pour générer des appels automatiques et simultanés, nous avons besoin d'un serveur pour émettre ces différents appels. Le serveur Trixbox joue le rôle d'IPBX et ce choix était imposé dans le cahier des charges du projet.

2.3 Les différents services du serveur Trixbox

2.3.1 Le service DHCP

Avant la réception du matériel des MMA, le fonctionnement du projet était basé sur deux Trixbox. Pour pouvoir utiliser des softs phones installés sur des ordinateurs, des Ipphones ou d'autres périphériques IP avec la Trixbox, il faut configurer un serveur DHCP.

2.3.1.a Qu'est-ce qu'un DHCP

Dynamic Host Configuration Protocol (DHCP) est un protocole réseau dont le rôle est d'assurer la configuration automatique des paramètres IP d'une station, notamment en lui affectant automatiquement une adresse IP et un masque de sous-réseau. DHCP peut aussi configurer l'adresse de la passerelle par défaut, ou des serveurs de noms DNS.

2.3.1.b Son fonctionnement

L'ordinateur équipé de carte réseau, mais dépourvu d'adresse IP, envoie par diffusion un datagramme (DHCP DISCOVER) qui s'adresse au port 67 de n'importe quel serveur à l'écoute sur ce port. Celui-ci comporte entre autres l'adresse physique (MAC) du client.

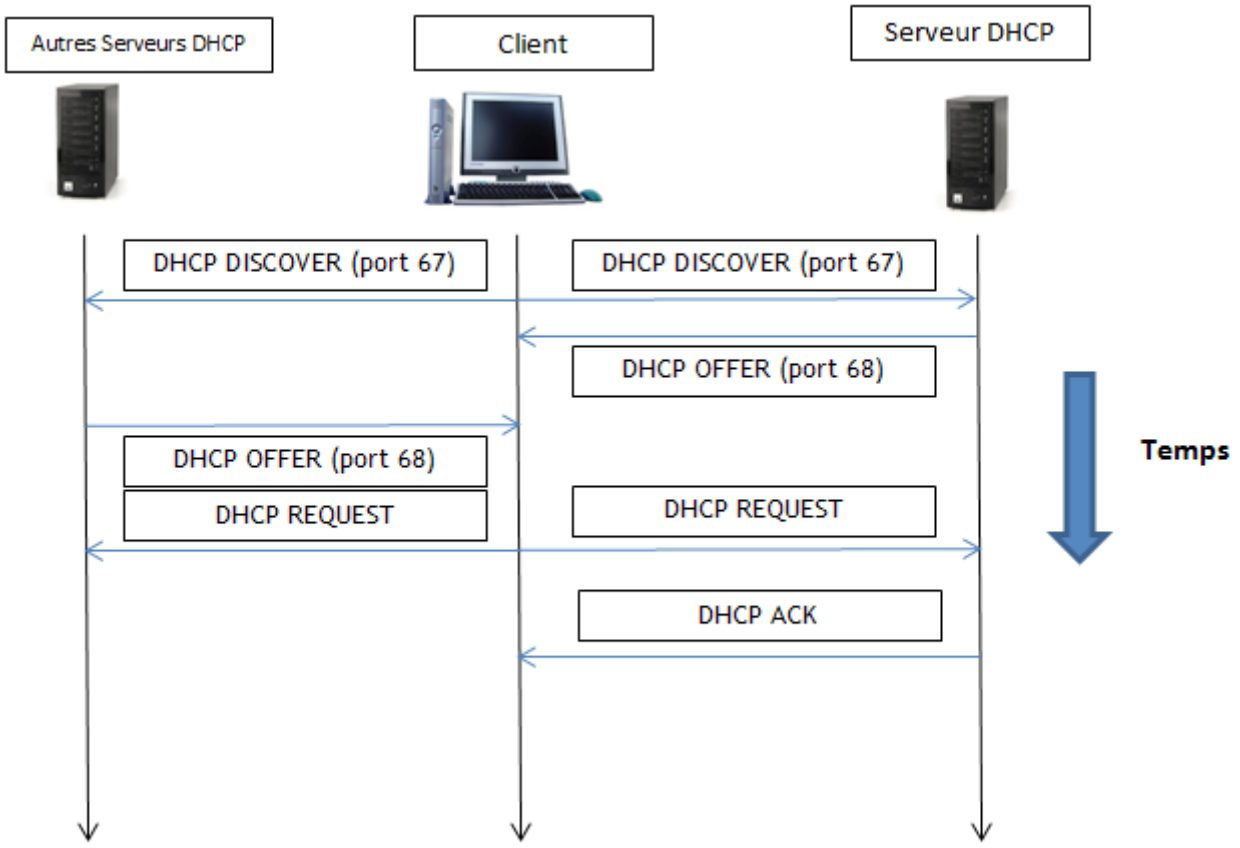
Tout serveur DHCP l'ayant reçu , s'il est en mesure de proposer une adresse sur le réseau auquel appartient le client, envoie une offre DHCP (DHCP OFFER) à l'attention du client (sur son port 68), identifié par son adresse physique. Cette offre comporte l'adresse IP du serveur, ainsi que l'adresse IP et le masque de sous-réseau qu'il propose au client. Il se peut que plusieurs offres soient adressées au client.

Le client retient une des offres reçues (la première qui lui parvient), et diffuse sur le réseau un datagramme de requête DHCP (DHCP REQUEST). Ce datagramme comporte l'adresse IP du serveur et celle qui vient d'être proposée au client. Elle a pour effet de demander au serveur choisi l'assignation de cette adresse, l'envoi éventuel des valeurs des paramètres, et d'informer les autres serveurs qui ont fait une offre qu'elle n'a pas été retenue.

Le serveur DHCP élabore un datagramme d'accusé de réception (DHCP ACK pour acknowledgement) qui assigne au client l'adresse IP et son masque de sous-réseau, la durée du bail de cette adresse (dont découlent deux valeurs T1 et T2 qui déterminent le comportement du client en fin de bail), et éventuellement d'autres paramètres :

- adresse IP de la passerelle par défaut,
- adresses IP des serveurs DNS,
- adresses IP des serveurs NBNS (WINS).

Les serveurs DHCP doivent être pourvus d'une adresse IP statique.



2.3.1.c Mise en place sur le serveur

Pour ce faire, nous devons accéder au fichier *dhcpd.conf* qui est situé dans */etc/dhcpd.conf* avec la commande *vi dhcpd.conf*

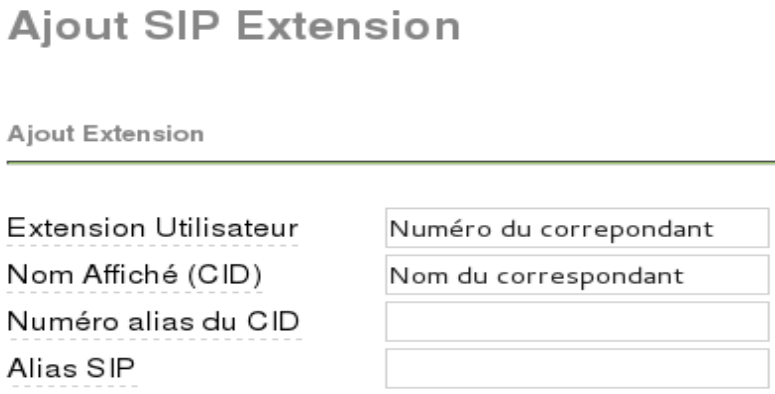
```
subnet 192.168.0.0 netmask 255.255.255.0 {

  option routers                  192.168.0.5;
  option subnet-mask              255.255.255.0;
  option nis-domain               "asterisk.local";
  option domain-name              "asterisk.local";
  option domain-name-servers      192.168.0.5;
  option time-offset              -18000; # Eastern Standard Time
  option ntp-servers              192.168.0.5;
  option tftp-server-name         "192.168.0.5";

  range dynamic-bootp 192.168.0.128 192.168.0.254;
  default-lease-time 43200;
  max-lease-time 86400;
}
~
```

Pour configurer le serveur DHCP, il faut définir la plage d'adresse IP dynamique. Dans notre cas, *192.168.0.128 à 192.168.0.254*. Dans un second temps, pour plus de facilité d'administration du système, une IP doit être rattachée à une adresse MAC comme ce ci:

host IPPhone
{ hardware ethernet 00:80:9F:6C:20:3A;
fixed-address 192.168.0.20; }

2.3.2 Administration web : Création de comptes SIP

Pour faire communiquer 2 softs-Phone entre eux, différents protocoles sont possibles. Le SIP est le plus simple à mettre en place, et permet un coût réduit au maximum.

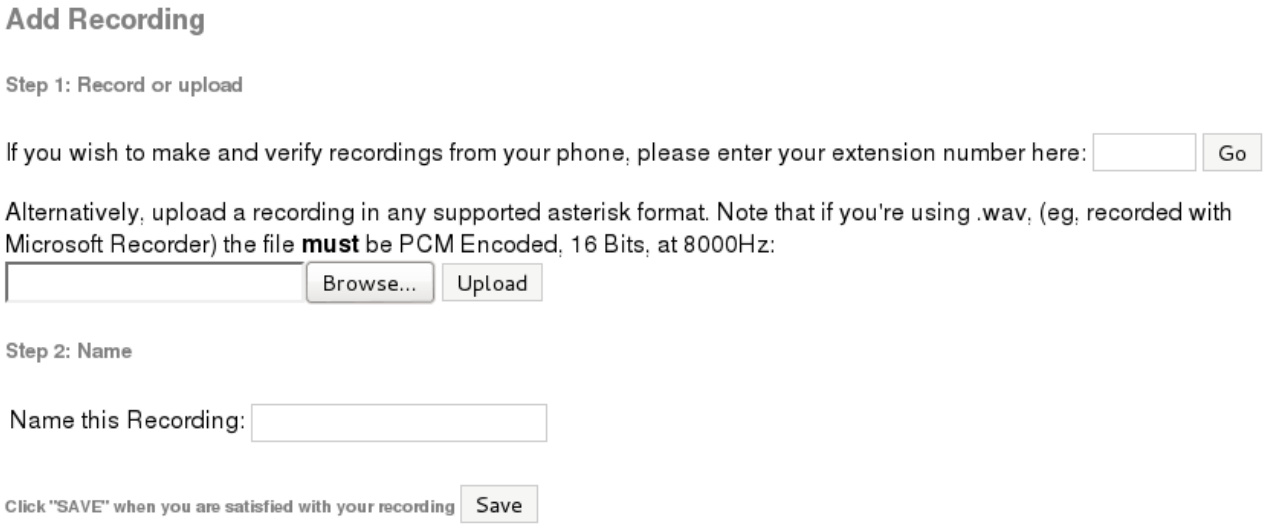


La création d'un compte SIP consiste à déclarer un nouvel usager. Cela lui permet de se connecter avec un soft-Phone sur le serveur.

Pour ce cela, nous devons déclarer son numéro qui correspond à son extension, son CID qui est le nom affiché lors de l'appel et son code secret.

2.3.3 Administration web : Enregistrement d'un guide

Dans un premier temps, il faut enregistrer les différents guides vocaux. Le guide a différents rôles. Le premier est d'accueillir par un message audio l'appelant sur la plate-forme téléphonique. Son second est de diriger le client dans les différents choix disponibles dans les menus. Nous devons pour cela aller dans les *réglages du PBX*[3] *(PABX)* puis dans le système *d'enregistrement*.

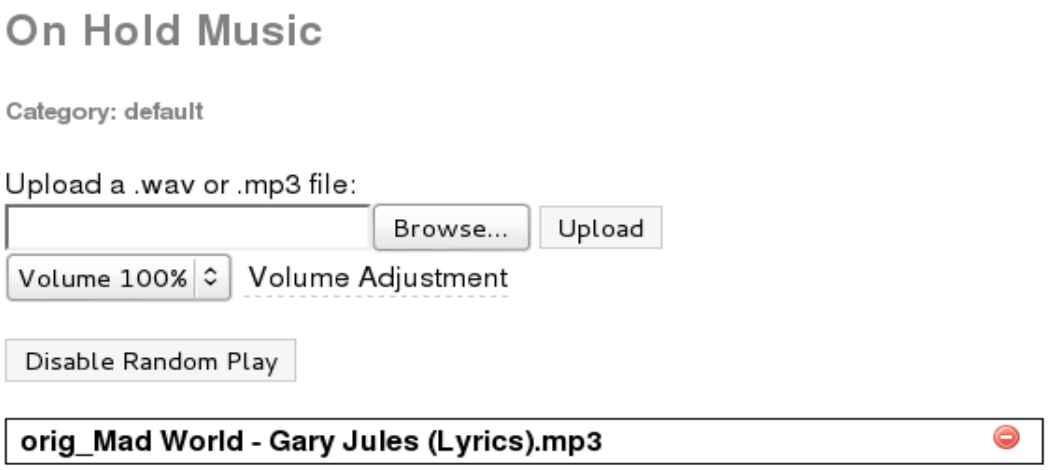

Add Recording
Step 1: Record or upload
If you wish to make and verify recordings from your phone, please enter your extension number here:
Go
Alternatively, upload a recording in any supported asterisk format. Note that if you're using .wav, (eg, recorded with Microsoft Recorder) the file must be PCM Encoded, 16 Bits, at 8000Hz:
Browse...
Upload
Step 2: Name
Name this Recording:
Click "SAVE" when you are satisfied with your recording
Save

3 PABX : Private Automatic Branch eXchange (Autocommutateur téléphonique privé)

Différents modes d'enregistrements sont possibles, soit par un fichier audio enregistré au format .wav (le .mp3 n'est pas conseillé mais fonctionne parfaitement) soit l'enregistrement via un poste (ou softPhone) connecté à la Trixbox. Pour cela, il suffit de préciser l'extension du poste qui effectuera l'enregistrement. En précisant le nom et en cliquant sur *save*, le guide est sauvegardé.

2.3.4 Administration web : Installation d'une musique d'attente

Pour accueillir les appels, sur le numéro principal ou pour faire patienter un appelant, nous devons créer des musiques d'attentes. Rendez-vous donc dans la section *music on hold*. Deux choix se présentent, soit utiliser la catégorie par défaut, soit en créer une nouvelle.

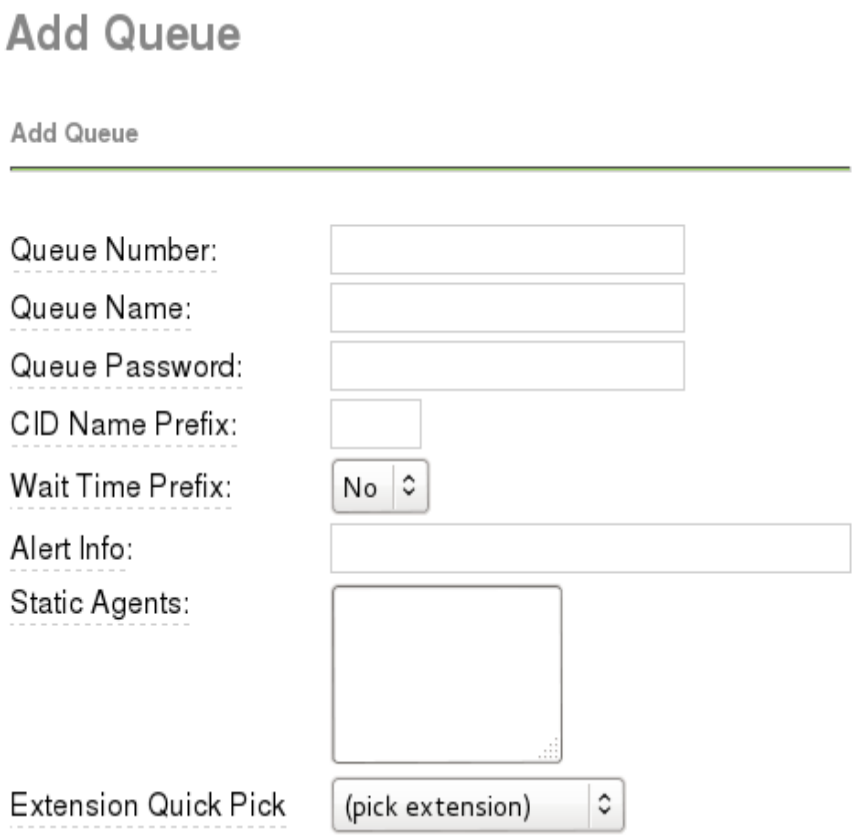


Par la suite, nous pourrons accéder à celles-ci via un menu déroulant, dans les files d'attente par exemple.

2.3.5 Administration web : Installation d'une file d'attente

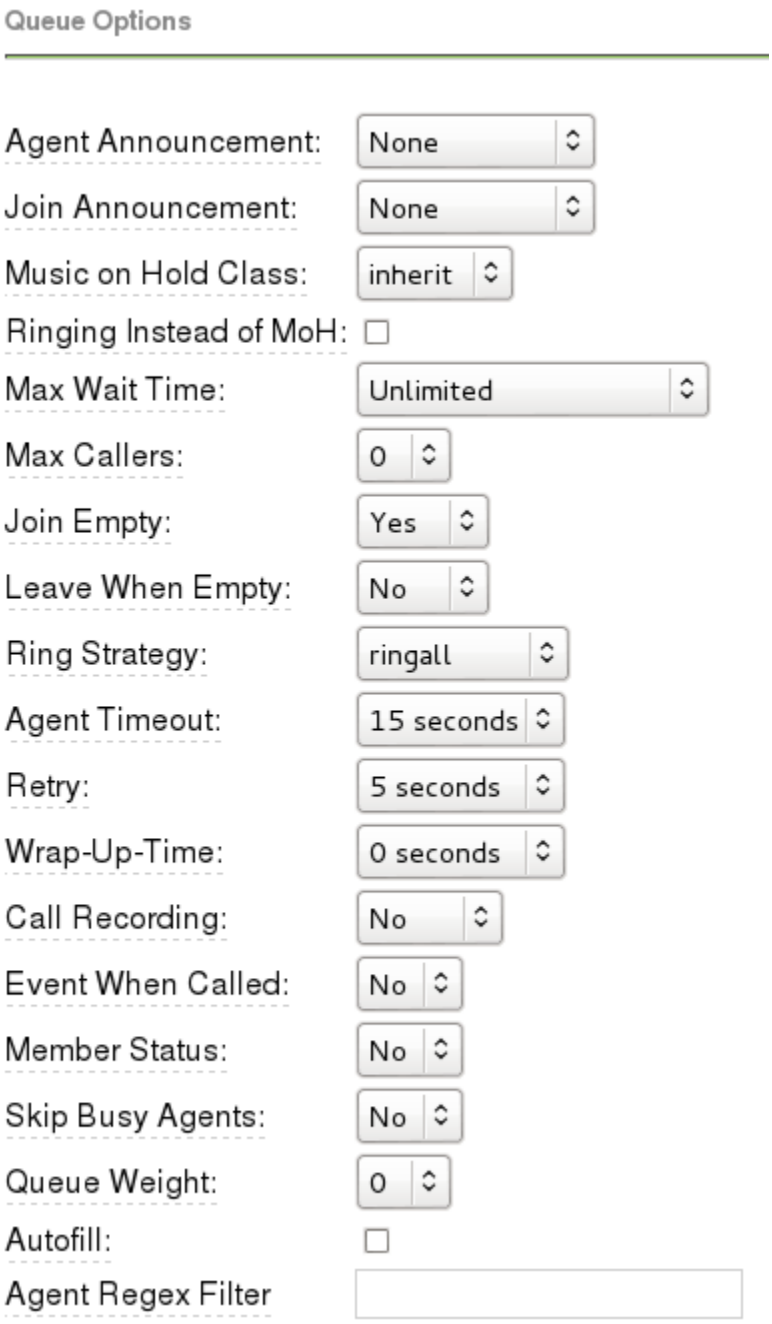
Dans un menu interactif, il est possible de rediriger un choix vers une file d'attente qui fera suite sur un groupe de traitement. Comment créer une nouvelle file ?

Dans ce but, nous devons aller dans les *réglages du PBX* puis dans *Queues*.



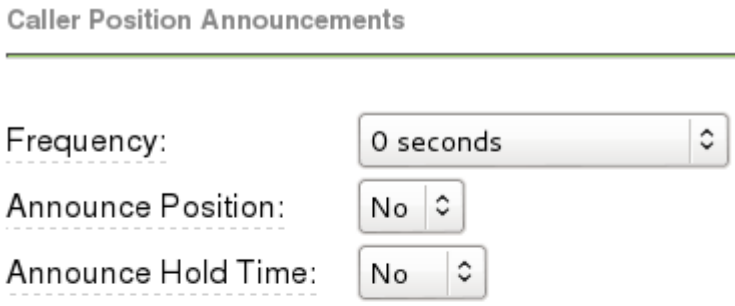
Dans cette première étape, différents champs sont à renseigner :

- Le numéro de la file
- Le nom de la file
- Un mot de passe (pas obligatoire)
- La liste des agents assignés au groupe de traitement
- Une destination finale au cas où le groupe de traitement serait indisponible.

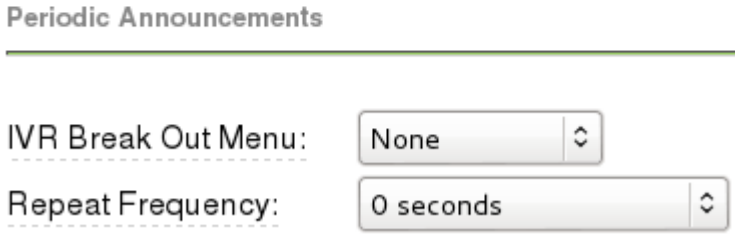


- Agent Announcement : C'est le message qui sera joué lorsque l'agent prend la communication.
- Joint Announcement : Il s'agit du message qui est entendu par l'appelant lorsqu'il entre dans la file d'attente.
- Music on Hold Class : Ici est renseigné le groupe de musique, qui sera jouée pendant l'attente.
- Max Wait Time : Temps d'attente maximum.
- Max Callers : Nombre maximum dans la file d'attente
- Ring strategy: Mode de sonnerie (dans notre cas sur tous les postes)
- Agent timeout : Durée avant non réponse
- Retry : Temporisation avant essai suivant
- WrapUp Time : Temps de repos entre deux appels

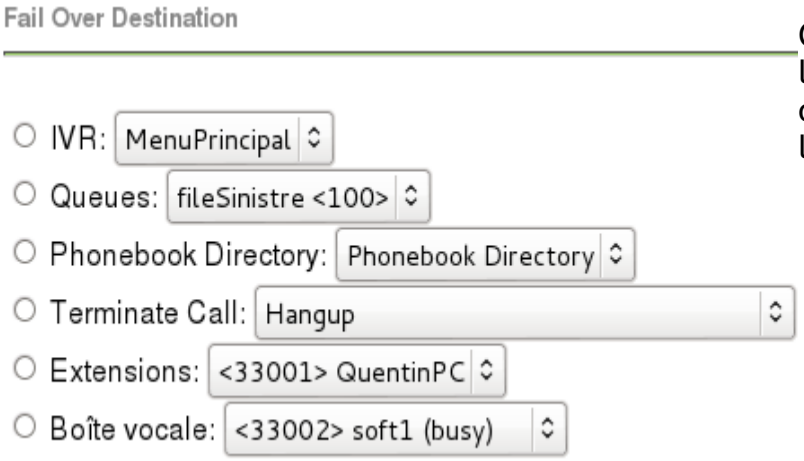
- Caal recording : enregistrement de la conversation
- Skip Busy Agents: évite les agents occupés



Cette section est dédiée aux annonces durant l'attente, on peut y régler la fréquence d'annonce en seconde, et choisir le type d'annonce, la position, le temps d'attente.



Cette deuxième catégorie est consacrée à l'annonce pour revenir au menu précédent.



Cette dernière partie, détermine le choix a faire en cas de time out dans la file d'attente. Soit l'appel sera redirigé vers :

- le RVI
- Une autre file d'attente
- Fin d'appel
- Un poste spécifié
- Une boite vocale

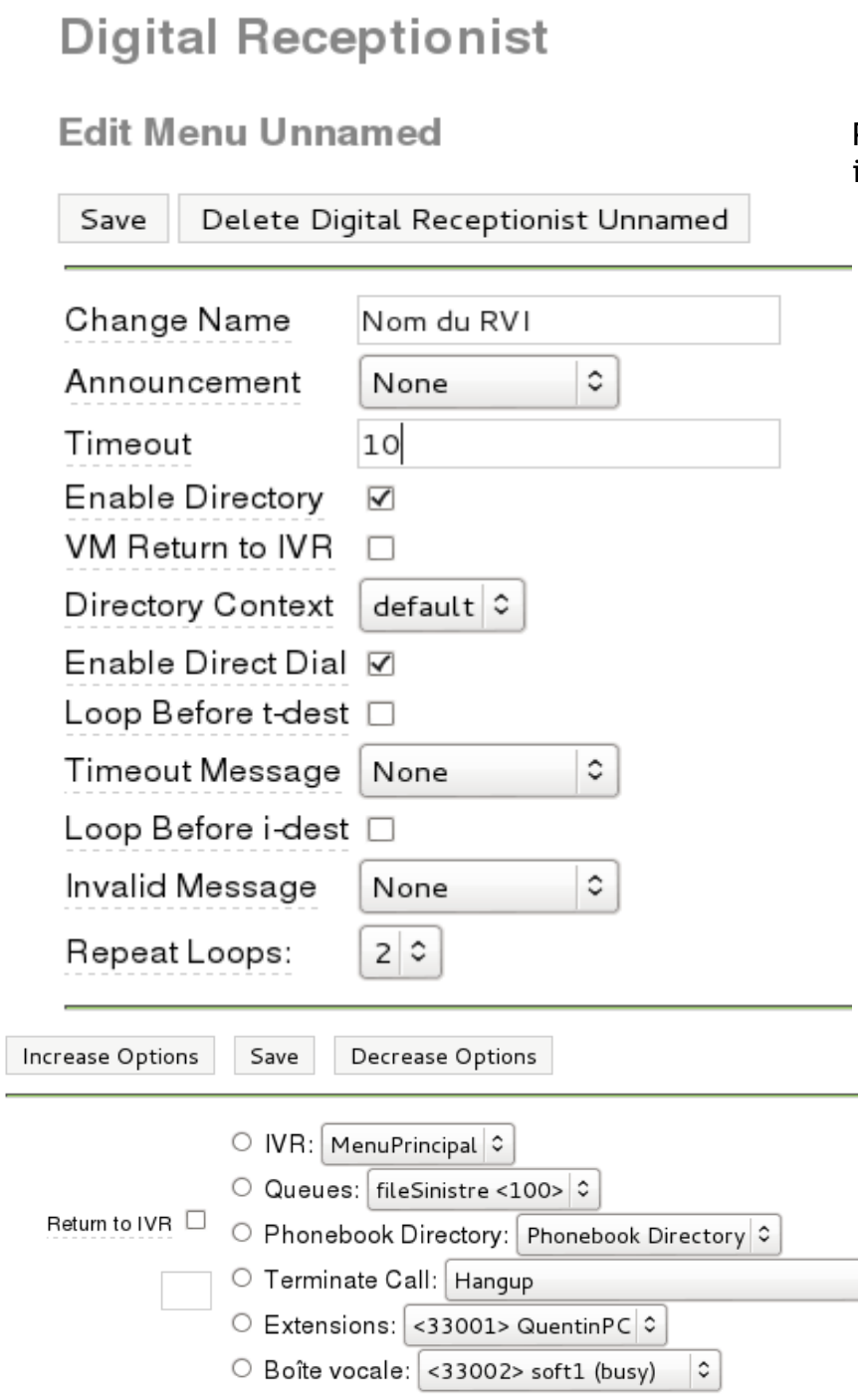
2.3.6 Administration web : Installation d'un IVR

2.3.6.a Qu'es qu'un IVR?

Interactive voice response, est une technologie qui permet à un ordinateur d'interagir avec les humains par l'utilisation de la voix et des tonalités DTMF à partir du clavier d'entrée.

2.3.6.b Configuration

La dernière étape consiste en la création du menu interactif. Il suffit de se rendre dans *IVR*.



Pour la création, quelques informations sont nécessaires :

- Nom du RVI
- Announcement :

  texte d'annonce
- Timeout : Temps avant le message de timeout
- TimeOut message : message en cas de non détection
- Invalid Message :

  Message en cas de touche invalide
- Repeat loop : nombre de répétitions de « Announcement »

Dans la seconde partie, nous pouvons définir les choix qui seront actifs dans le menu, comme choisir un deuxième menu, une file d'attente, un poste, ...

2.3.7 Le protocole FTP

Pour pouvoir exécuter les scripts qui nous permettront de lancer les différents appels, nous devons installer un serveur FTP sur le quelle nous enverrons les fichiers .call à distance. Dans un second temps il servira aussi à modifier le fichier *extension.conf* qui gère les différents scénarios d'appels.

2.3.7.a *Qu'est-ce qu'un FTP*

File Transfer Protocol (protocole de transfert de fichiers), ou FTP est un protocole de communication destiné à l'échange informatique de fichiers sur un réseau TCP/IP.

Il permet, depuis un ordinateur, de copier des fichiers vers un autre ordinateur du réseau, ou encore de supprimer ou de modifier des fichiers sur cet ordinateur. Ce mécanisme de copie est souvent utilisé pour alimenter un site web hébergé chez un tiers.

Sa variante est protégée par les protocoles SSL ou TLS (SSL étant le prédécesseur de TLS) s'appelle FTPS. Il appartient à la couche application du modèle OSI , utilise une connexion TCP.

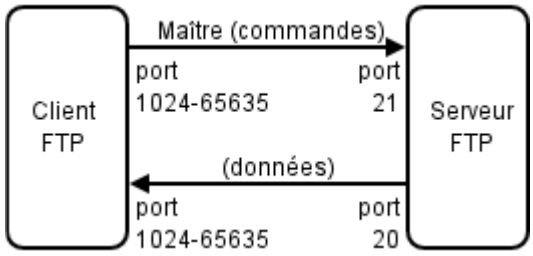
2.3.7.b Son Fonctionnement

Le protocole utilise deux types de connexions TCP :

- Une connexion de *contrôle* initialisée par le client, vers le serveur (port 21 en général), pour transmettre les commandes de fichiers (transfert, suppression de fichiers, renommage, liste des fichiers…).
- Une connexion de *données* initialisée par le client ou le serveur pour transférer les données requises (contenu des fichiers, liste de fichiers).

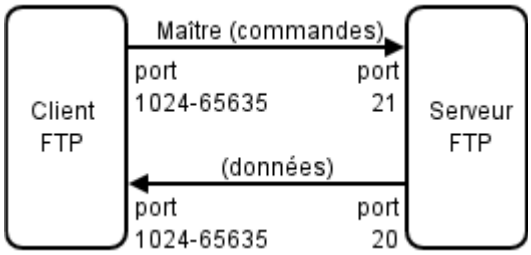
Et il peut s'utiliser de deux façons différentes.

En mode passif :



En mode passif, le serveur FTP détermine lui-même le port de connexion à utiliser pour permettre le transfert des données (data connexion) et le communique au client.

En mode actif :



En mode actif, c'est le client FTP qui détermine le port de connexion à utiliser pour permettre le transfert des données. Ainsi, pour que l'échange des données puisse se faire, le serveur FTP initialisera la connexion de son port de données (port 20) vers le port spécifié par le client.

2.3.8 PhpMyAdmin

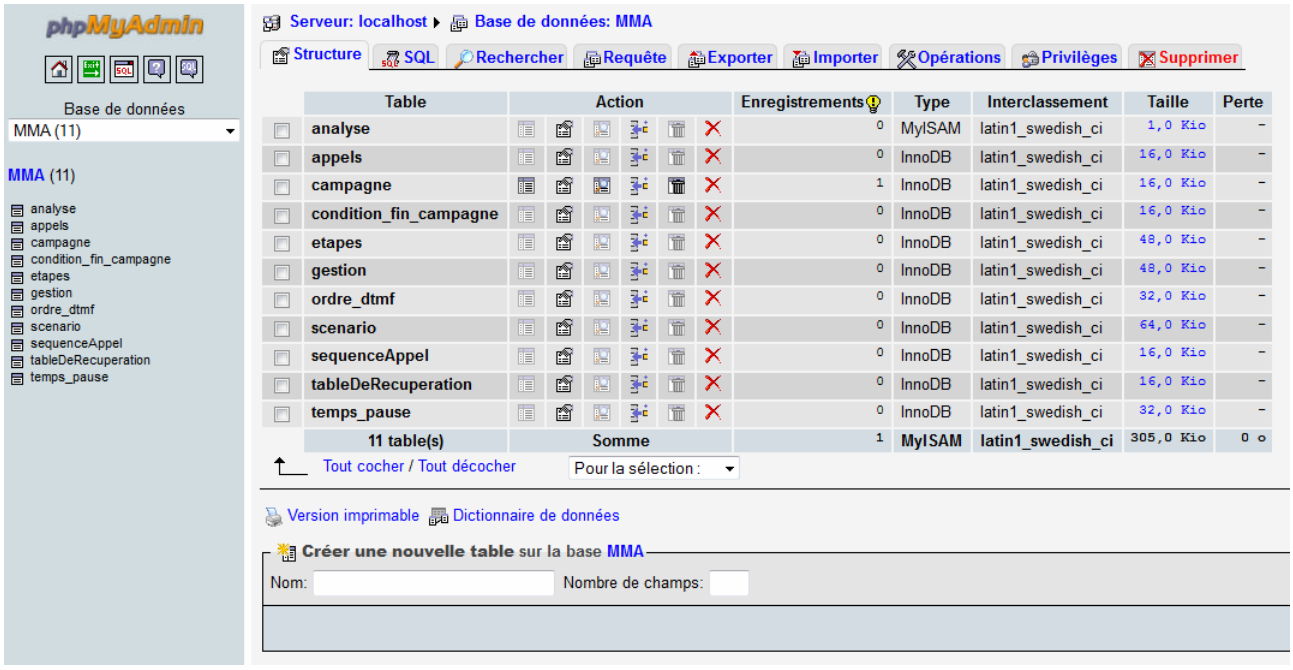
Dans le cadre de la création et du suivi des campagnes d'appels, nous devons sauvegarder les différentes données. Pour cela nous avons choisi d'installer phpMyAdmin, qui nous permettra, de la création, jusqu'à la gestion et de stocker ces donnée dans une base MYSQL.

2.3.8.a Qu'est-ce que phpMyAdmin ?

phpMyAdmin est une application Web de gestion pour les systèmes de gestion de base de données MySQL réalisée en PHP et distribuée sous licence GNU GPL[4].

Cette interface pratique permet d'exécuter, très facilement de nombreuses requêtes comme la création de tables de données, des insertions, des mises à jour …

Ce système est très pratique pour sauvegarder une base de données sous forme de fichier .sql et ainsi transférer facilement ses données. De plus celui-ci accepte la formulation de requêtes SQL directement en langage SQL.



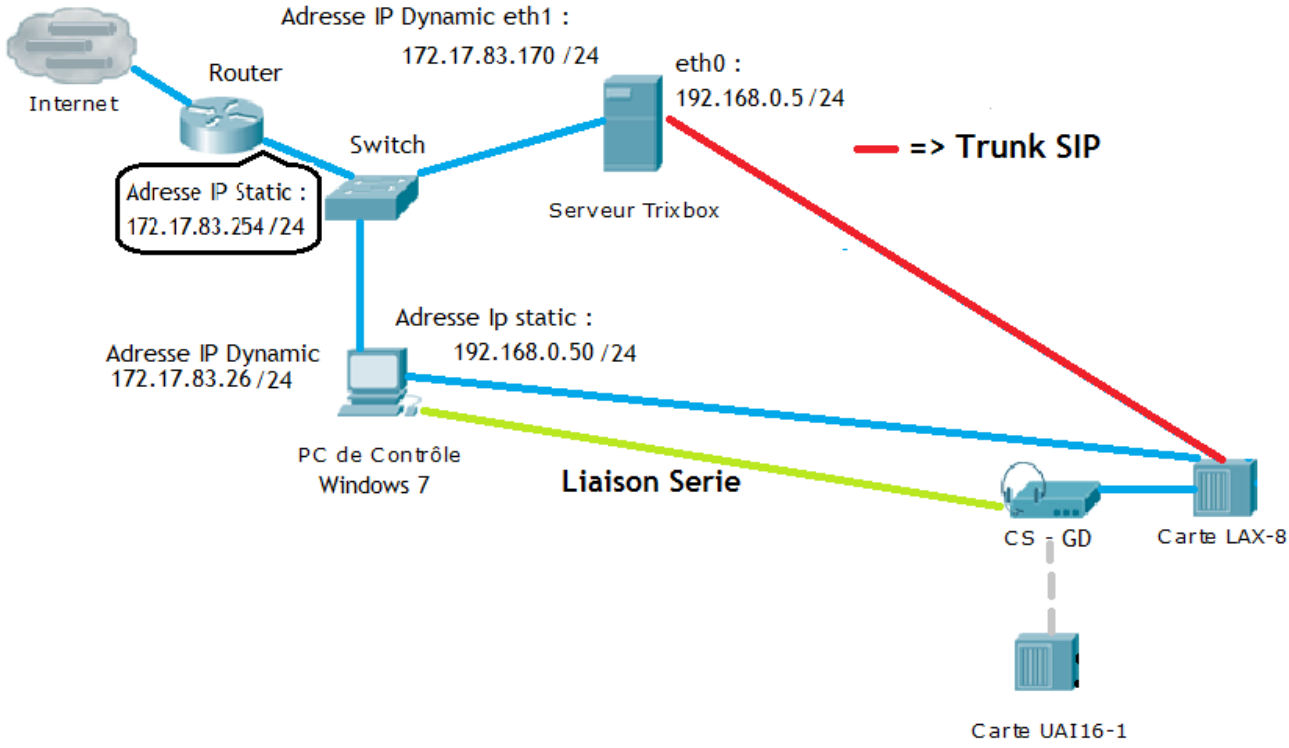
4 Licence publique générale, fixe les conditions légales de distribution des logiciels libres du projet GNU.

2.4 Étude d'un Trunck SIP

2.4.1 Qu'est-ce qu'un Trunk SIP ?

SIP est un protocole VoIP (Voice over Internet Protocol), il est utilisé par les fournisseurs de services de téléphonie Internet (ITSP[5]) fournissent des services de communications unifiées pour les clients équipés d'un autocommutateur SIP.

Dans une autre utilisation, qui sera notre cas dans ce projet, nous allons en avoir besoin pour faire le lien entre notre serveur Trixbox et L'OXE Alcatel pour pouvoir faire transiter des appels SIP de l'un vers l'autre.

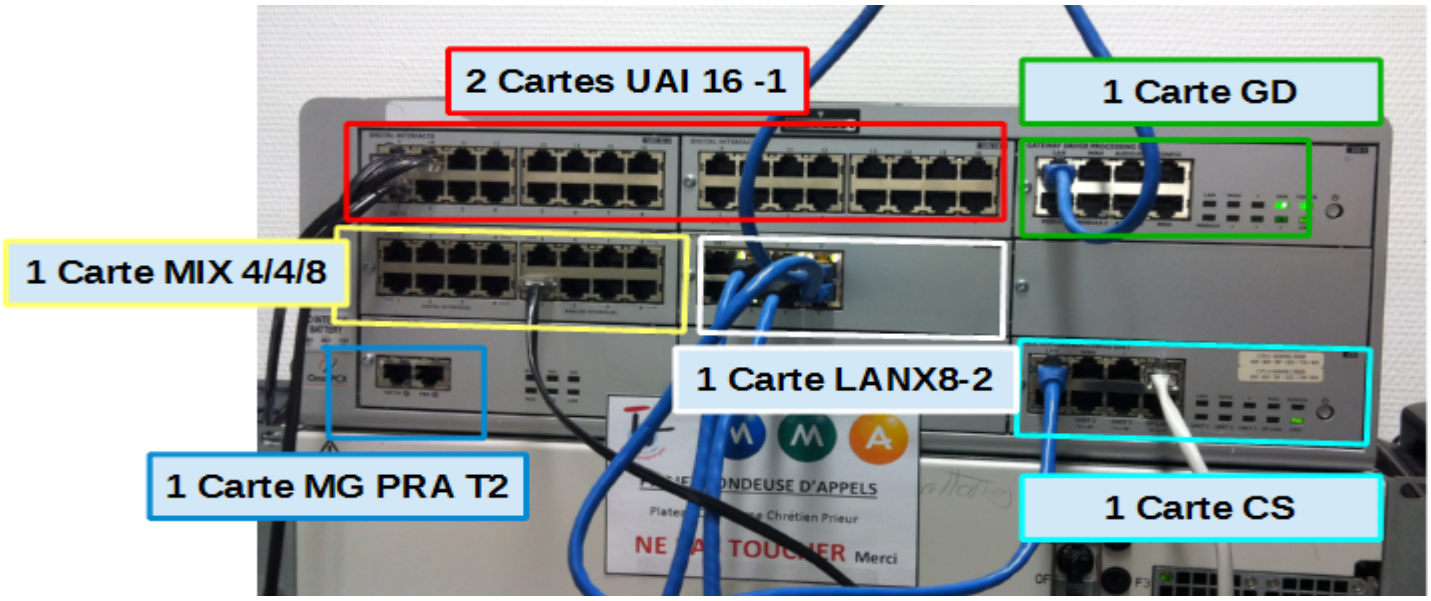


2.5 Mise en place de la maquette MMA: Partie technique

Dans la partie commune, nous avons décrit l'installation du centre d'appels, prêtée par les MMA. Nous allons maintenant nous intéresser au côté technique de cette mise en place.

5 Internet telephony service provider

2.5.1 Visuelle de la maquette



Les différents composants qui forment la maquette sont :

- 2 Cartes UAI 16-1 :
  - Carte pour 16 équipements numériques (ex : poste téléphonique)
- 1 Carte MIX 4/4/8, elle comporte :
  - 4 accès de type T0
  - 4 ports numériques
  - 8 ports analogiques
- 1 Carte LANX8-2 :
  - Il s'agit d'un switch intégré.
- 1 carte MG PRA T2 :
  - Comporte 2 accès de types T2
- 1 Carte GD :
  - Détiens toute intelligence, se place en position 0 du rack.
- 1 Carte CS :
  - Appelé aussi Call Server
  - Elle est connectée à la GD, est administrée via telnet et supervisée via CCSupervision

2.5.2 Première installation

1. La première étape consiste à exécuter la commande *netadmin* via une connexion telnet sur l'adresse IP de la CS *192.168.0.10*. Elle est utilisée pour configurer l'adresse IP ainsi que les différents paramètres réseau.
2. La seconde étape a pour but de créer une base de données vierge, pour y sauvegarder toute la configuration de la CS, il faut passer par le compte *swinst*.

   À partir de cet instant, le service de téléphonie est fonctionnel
3. Il faut maintenant configurer la GD (Geteway Driver). Au premier démarrage, il faut lui renseigner dans sa configuration, via le port console, son adresse IP, le masque sous réseau utilisé et l'adresse IP de la CS
4. Via le script «mgr», sur la CS nous allons créer l'alvéole, qui est de type 9 cartes autrement appelées «media geteway 9 positions». Dans les paramètres Ethernet, il faut insérer l'adresse physique de la GD dans la CS.
5. Il ne reste plus qu'à déclarer les autres cartes via le même script *mgr* sur la CS.

2.5.3 Choix du protocole

Nous avons eu besoin de choisir parmi plusieurs protocoles afin d'assurer la liaison entre la trixbox et l'Alcatel OMNI-PCX ENTERPRISE.

Voici une étude des différents protocoles que nous avons sélectionnés pour les comparer.

Session Initiation Protocol (SIP) est un protocole standard ouvert de gestion de sessions souvent utilisé dans les télécommunications multimédias (son, image, etc.). Il est depuis 2007 le plus courant pour la téléphonie par internet (la VoIP).

H.323 regroupe un ensemble de protocoles de communication de la voix, de l'image et de données sur IP. C'est un protocole défini comme : « Systèmes de communication multimédia en mode paquet ».

Media Gateway Control Protocol (MGCP) est un protocole permettant de contrôler les passerelles multimédias (Media Gaeway) qui assurent la conversion de la voix et de la vidéo entre les réseaux IP et le réseau téléphonique commuté (RTC).

Choix du protocole :

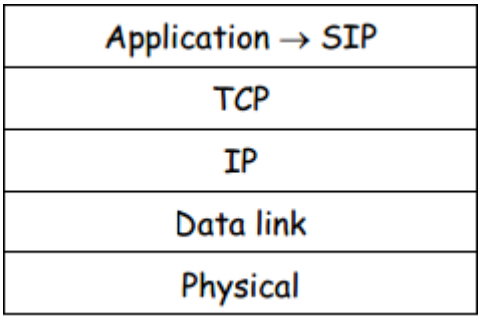
Comparatifs des protocoles principaux :

	SIP	H323	MGCP
Complexité	Faible (Simple par sa nature ex : Http)	Élevée(nécessite un compilateur)	Élevée(pour la téléphonie moderne)
Protocoles transports	TCP ou UDP	TCP ou UDP	TCP ou UDP
Avantages	Compatible avec plusieurs technologies	Beaucoup utilisé	Conçu pour les opérateurs voulant faire du RTC-IP-RTC
Inconvénients	Sécurité	-Difficultés avec les firewall -Manque de comptabilité	Service supplémentaire de téléphonie inexistant
Exemple de solutions utilisant le protocole	Yahoo !, MSN Messenger...	Livecom	Les entreprises

Nous avons choisi le protocole SIP pour sa complexité moindre (Simple par sa nature textuelle à l'exemple de HTTP) ainsi que ses principaux avantages, mais aussi par contrainte de matériel (nous ne possédons pas la carte requise sur notre OMNI-PCX pour utiliser le protocole H323).

Nous ne disposions pas de la carte requise sur le PABX pour utiliser le protocole H323.

L'**architecture en couches de SIP**, telle que la présente le modèle OSI, fait apparaître une palette de nombreux protocoles



- **RSVP** est un protocole utilisé pour réserver les ressources réseau sur IP avec une excellente qualité de service (QoS)
- **R.T.P.** (Real-time Transport Protocol) pour transporter des informations en temps réel avec une excellente qualité de services
- **R.T.C.P.** (Real-Time streaming Control Protocol) pour assurer le contrôle de flux des données multimédia
- **S.A.P.** (Session Announcement Protocol) pour préciser si les sessions multimédias ouvertes le sont en multicast
- **S.D.P.** (Session Description Protocol) est un protocole de description des sessions multimédia.

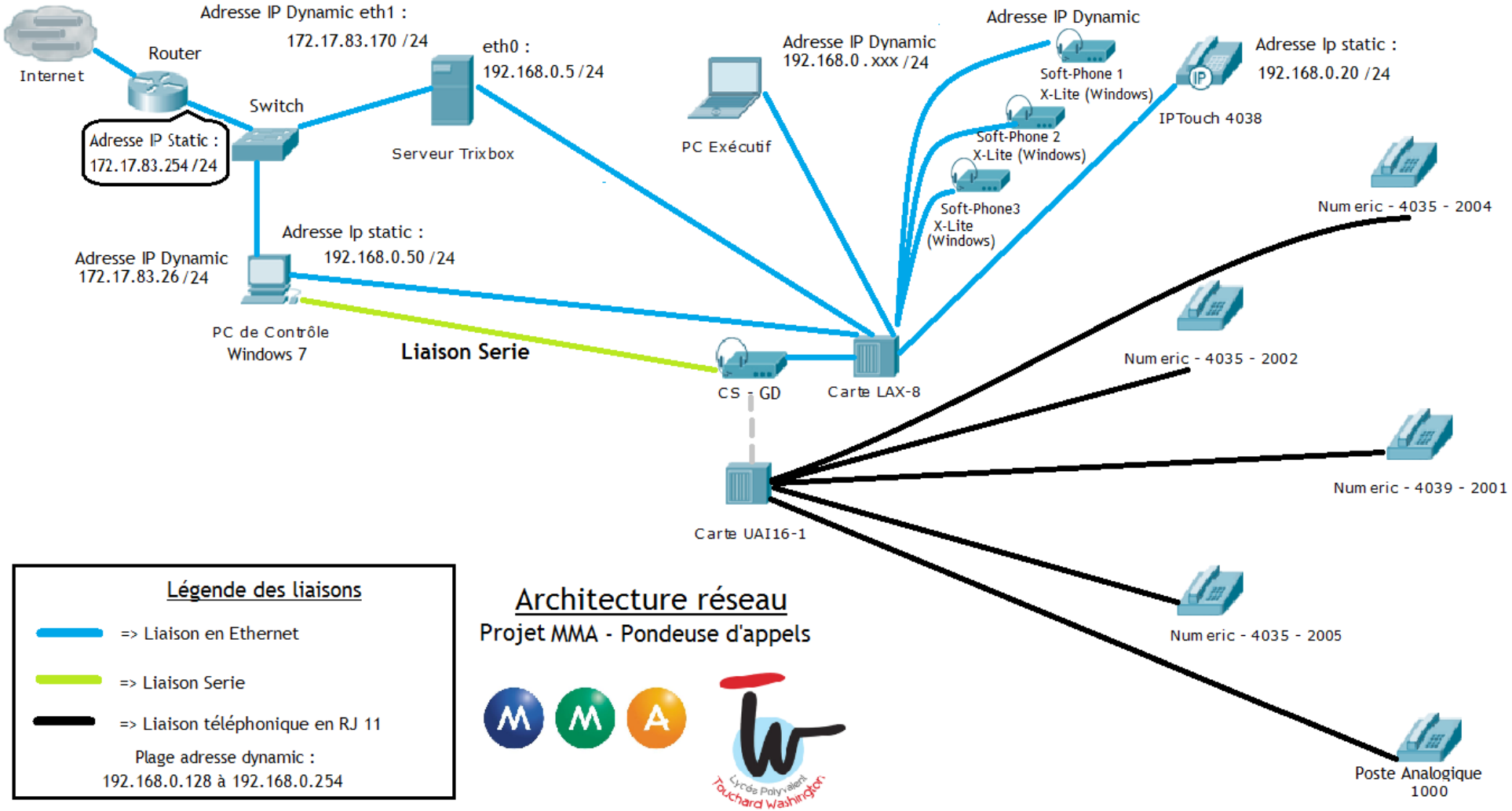
3. Conception préliminaire : Partie Plateau Mathias (E1)

3.1 L'architecture

Je vais me trouver pendant la durée du projet du côté de la pondeuse d'appels.Les N appels simultanés seront envoyés via faisceaux SIP (voir choix du protocole). Le centre d'appels sera géré par Quentin PRIEUR (étudiant 4).

La trixbox et la machine contenant l'application communiquerons via un cable ethernet, La trixbox et l'Alcatel OMNI-PCX communiquent via faisceau SIP (trunk) : un lien TRUNK est un lien qui permet de faire transiter plusieurs VLAN sur un seul lien physique.

Ci-dessous, voici un plan de l'architecture réseau de notre système.


Internet
Router
Adresse IP Dynamic eth1 :
172.17.83.170 /24
Switch
Adresse IP Static :
172.17.83.254 /24
eth0 :
192.168.0.5 /24
Serveur Trixbox
PC Exécutif
Adresse IP Dynamic
192.168.0 . xxx /24
Adresse IP Dynamic
Soft-Phone 1
X-Lite (Windows)
Soft-Phone 2
X-Lite (Windows)
Soft-Phone3
X-Lite
(Windows)
IPTouch 4038
Adresse Ip static :
192.168.0.20 /24
Num eric - 4035 - 2004
Num eric - 4035 - 2002
Num eric - 4039 - 2001
Num eric - 4035 - 2005
Poste Analogique
1000
Adresse Ip static :
192.168.0.50 /24
Adresse IP Dynamic
172.17.83.26 /24
PC de Contrôle
Windows 7
Liaison Serie
CS - GD
Carte LAX-8
Carte UAI16-1
Légende des liaisons
=> Liaison en Ethernet
=> Liaison Serie
=> Liaison téléphonique en RJ 11
Plage adresse dynamic :
192.168.0.128 à 192.168.0.254
Architecture réseau
Projet MMA - Pondeuse d'appels
Lycée Polyvalent
Touchard Washington

3.2 Architecture logicielle :

Pour l'étude préliminaire, nous avons eu besoin de multiples logiciels qui ont été installés sur les postes de travail :

- Réalisations des diagrammes : Modelio 2.2
- Développement des IHM et codage : QtCreator (version Linux et Windows 7).

3.2.1 Choix de la base de données

Le PABX Alcatel-Lucent Omni PCX Enterprise R8.0 possède en effet un système de gestion de base de données de type MAO qui utilise le protocole SQL. Or,si un jour le groupe MMA décide de changer son matériel, nous avons opté pour une autre solution en nous orientant vers une base de données MYSQL qui sera incorporée à notre système Trixbox, nous y accéderons via des requêtes SQL.

Version de la trixbox : 2.8.0.4

3.2.2 Les couches logicielles

Couches logicielles : CentOS, Asterisk , Trixbox version 2.8.0.4

Le serveur est installé sur un système CentOS , système de distribution linux, sur lequel est installé Asterisk. Asterisk est un autocommutateur téléphonie privée (PABX), il utilise 2 systèmes de licences. l'une libre et l'autre propriétaire pour systèmes GNU/Linux. Il permet, entre autres, la messagerie vocale, les files d'attente, les agents d'appels, les musiques d'attente , les mises en garde d'appels et la distribution des appels. Il est possible également d'ajouter l'utilisation des conférences par le biais de l'installation de modules supplémentaires et la recompilation des binaires.

Asterisk implémente les protocoles H323, SIP.

La trixbox est l'interface graphique d'Asterisk.

3.2.3 Choix du protocole

Nous avons eu besoin de choisir parmi plusieurs protocoles afin d'assurer la liaison entre la trixbox et l'Alcatel OMNI-PCX ENTERPRISE nous avons donc fais le choix de ce protocole avec l'étudiant 1.

Voici une étude des différents protocoles que nous avons sélectionné pour les comparer.

Session Initiation Protocol (SIP) est un protocole standard ouvert de gestion de sessions souvent utilisé dans les télécommunications multimédias (son, image, etc.). Il est depuis 2007 le plus courant pour la téléphonie par internet (la VoIP) .

H.323 regroupe un ensemble de protocoles de communication de la voix, de l'image et de données sur IP. C'est un protocole défini comme : « Systèmes de communication multimédia en mode paquet ».

Media Gateway Control Protocol (MGCP) est un protocole permettant de contrôler les passerelles multimédias (Media Gaeway) qui assurent la conversion de la voix et de la vidéo entre les réseaux IP et le réseau téléphonique commuté (RTC).

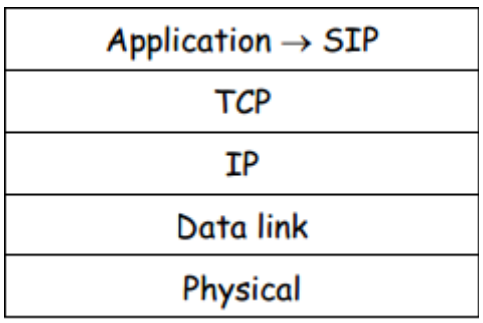
Choix du protocole :

Comparatifs des protocoles principaux :

	SIP	H323	MGCP
Complexité	Faible (Simple par sa nature ex : Http)	Élevée(nécessite un compilateur)	Élevée(pour la téléphonie moderne)
Protocoles transports	TCP ou UDP	TCP ou UDP	TCP ou UDP
Avantages	Compatible avec plusieurs technologies	Beaucoup utilisé	Conçu pour les opérateurs voulant faire du RTC-IP-RTC
Inconvénients	Sécurité	-Difficultés avec les firewall -Manque de comptabilité	Service supplémentaire de téléphonie inexistant
Exemple de solutions utilisant le protocole	Yahoo !, MSN Messenger...	Livecom	Les entreprises

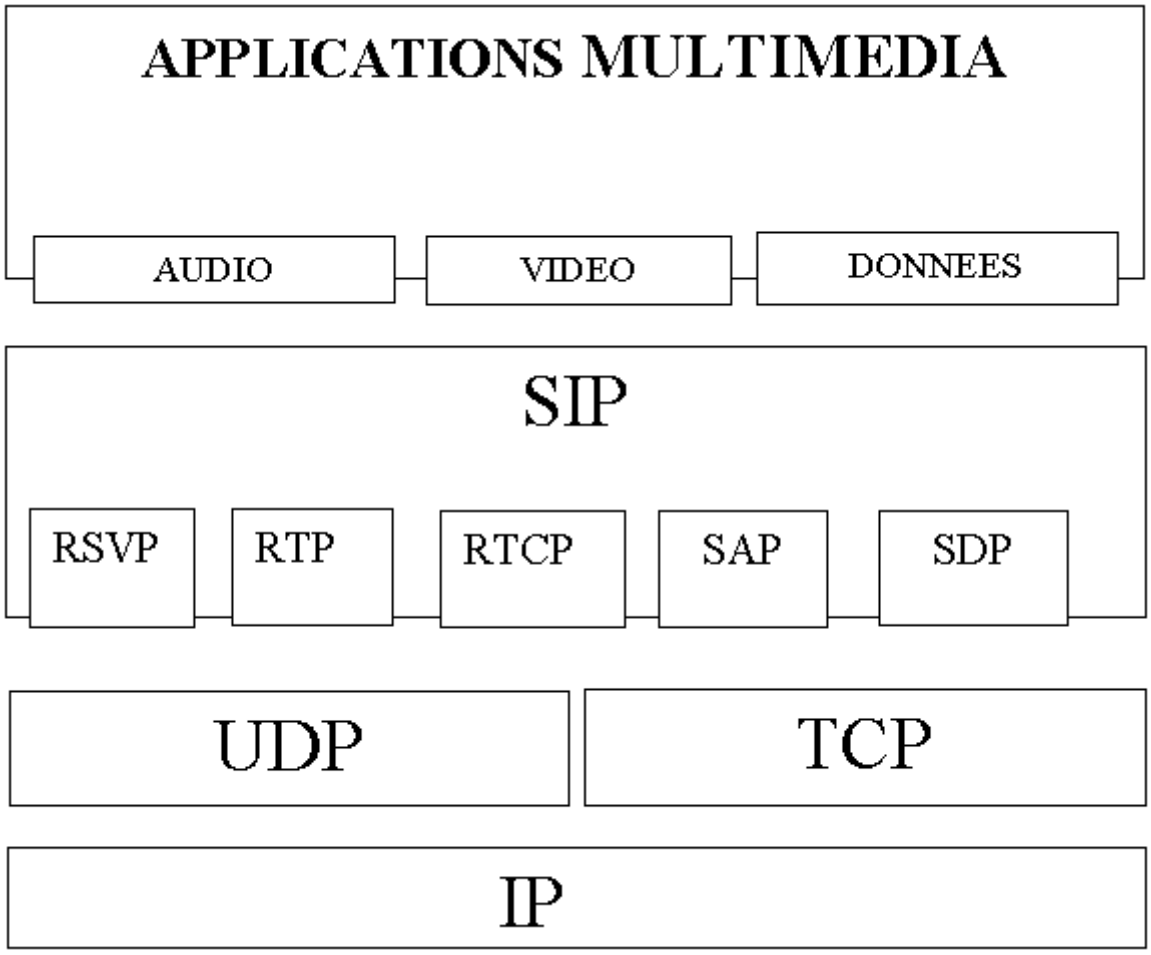
Nous avons choisi le protocole SIP pour sa complexité moindre (Simple par sa nature textuelle à l'exemple de Http) ainsi que ses principaux avantages, mais aussi par contrainte de matériel (nous ne possédons pas la carte requise sur notre OMNI-PCX pour utiliser le protocole H323).

L'**architecture en couches de SIP**, telle que la présente le modèle OSI, fait apparaître une palette de nombreux protocoles



:-**RSVP** est un protocole utilisé pour réserver les ressources réseaux sur IP avec une

excellente qualité de service(QoS) ;
-**R.T.P.**(Real-time Transport Protocol) pour transporter des informations en temps réel avec une excellente qualité de services;
-**R.T.C.P.**(Real-Time streaming Control Protocol) pour assurer le contrôle de flux des données multimédia;
-**S.A.P.**(Session Announcement Protocol) pour préciser si les sessions multimédia ouvertes le sont en multicast ;
-**S.D.P.**(Session Description Protocol) est un protocole de description des sessions multimédia.



3.3 Mon rôle au sein du projet

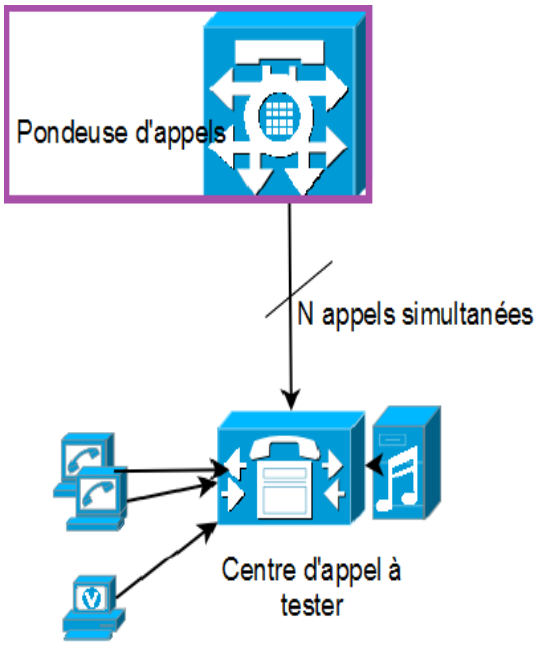
Le rôle du testeur d'appels est d'effectuer une campagne d'appels composée de un ou plusieurs scénarios, ces scénarios eux-mêmes composés d'étapes, ces étapes étant : les

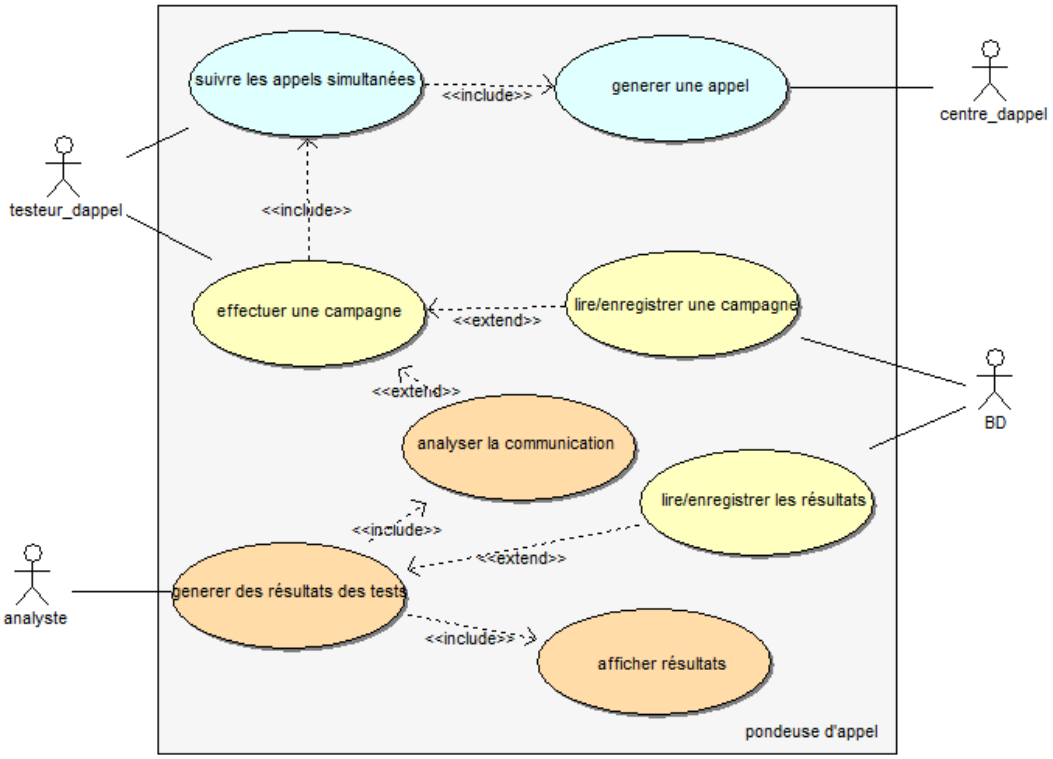
pauses et les différents DTMF (note : DTFM = dual-tone multi-frequency) de 0 à 9 ainsi que les caractères # et * et raccrocher.

Le DTMF est un système permettant la transmission de la numérotation dans un réseau à commutation de circuit classique.

L'appel est considéré lorsque le téléphone est décroché,toutefois, il n'est pas considéré comme une étape.

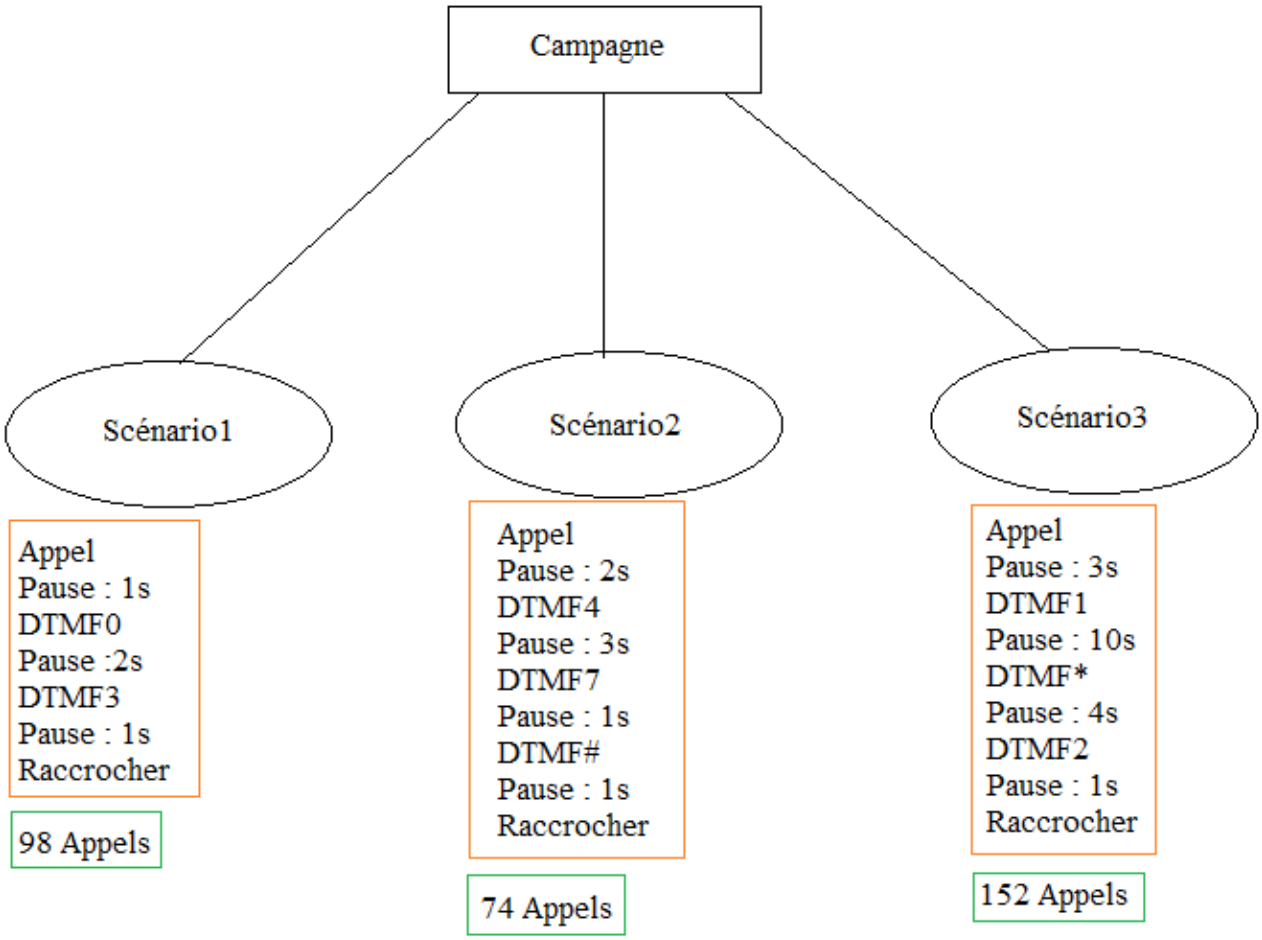
Étudiant 1





Une campagne est unique et est composée de un ou de plusieurs scénarios, pour une campagne nous avons un numéro d'appel (Exemple : 6000) et le temps de la campagne.

Un scénario est constitué de plusieurs étapes qui sont des couples temps de pause-DTMF et ce finit par raccrocher.



3.4 Les interfaces homme/ machine

En cas de nouvelle campagne l'utilisateur appuis sur le bouton Nouvelle campagne et entre premièrement le nom de la campagne.

Si l'utilisateur choisi d'accéder aux anciennes campagnes qui ont été enregistrées, il le choisi alors sur l'IHM , il peut aussi les supprimer si il appuis sur le bouton supprimer une campagne.

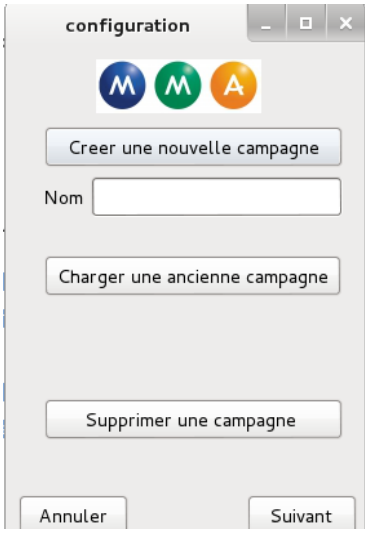
Le système accède alors à a la base de données,qui donne alors les campagnes(si il y en a) qui ont été enregistrées,elles sont donc affichées dans l'IHM, l'utilisateur peut maintenant en choisir une.

La campagne choisis est donc composée de Nscénarios qui sont eux même composés de Nétapes.

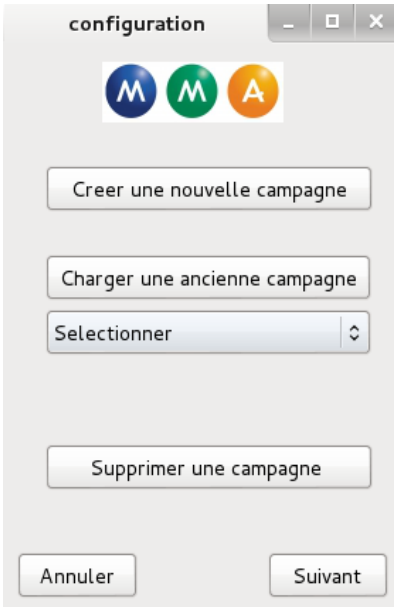
Une fois la campagne choisie, l'envoi des informations lié a la campagne est lancé.

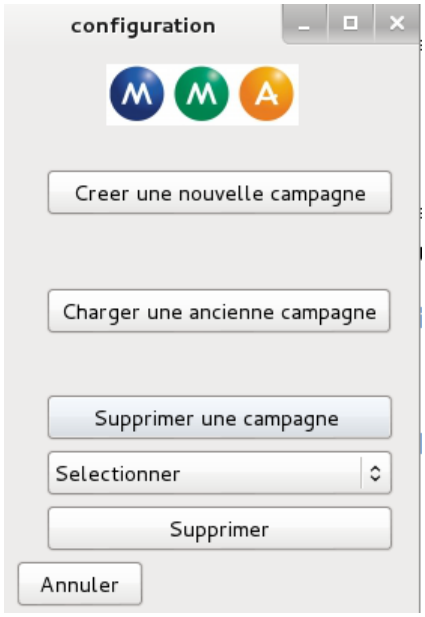
Suivant permet d'accéder à l'IHM suivante :

Nouvelle campagne:



Charger ou supprimer une ancienne campagne:





L'utilisateur choisit d'abord le numéro à appeler puis la durée de la session d'appels (Par défaut 1sec, maximum 1800sec).

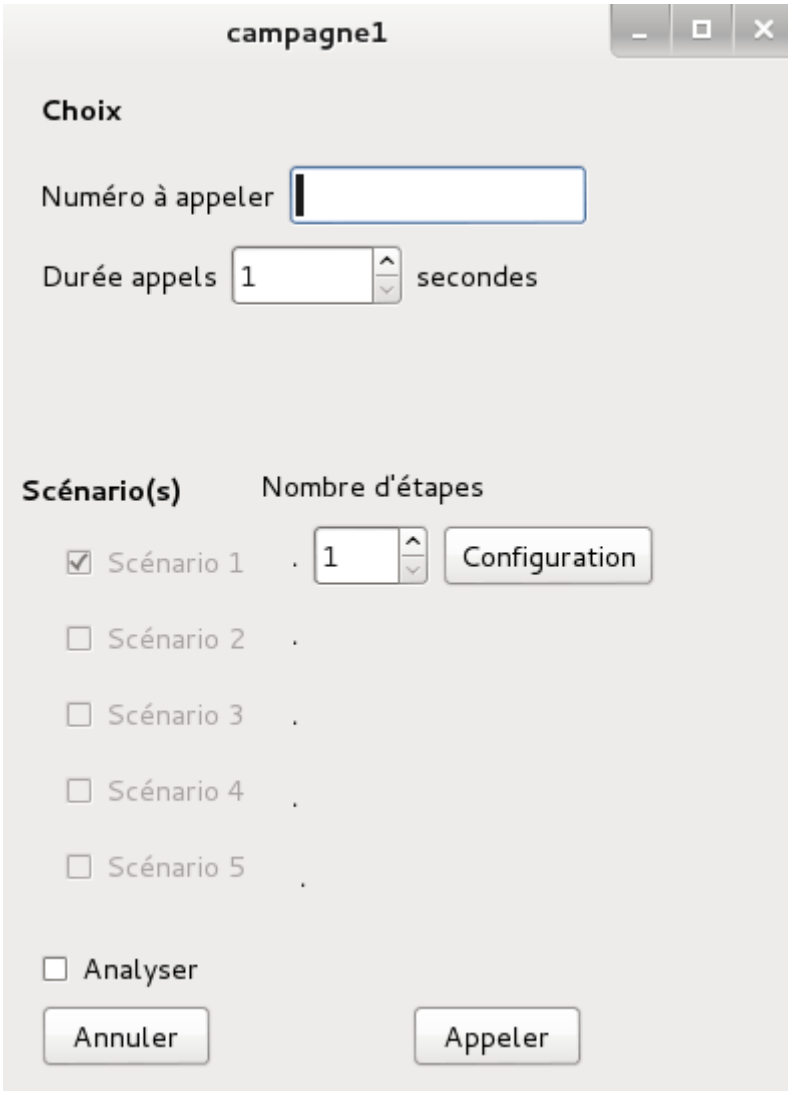
L'utilisateur choisis le nombre de scénarios souhaités puis pourra les configurer, 1 est coché au départ il peut en choisir 5 maximum.

Pour chacun des scénarios, un nombre d'étapes doit être indiqué.

Par défaut il y a un appel avec une étape, à l'intérieur un DTMF0 et un temps de pause de 0sec en cas d'appuis sur le bouton appeler sans paramétrage du scénario, ce sera donc le DTMF0 envoyé une fois durant l'appel.

Coché la QCheckbox analyser enverra sur l'IHM de l'étudiant 3 (Florian DAUVERGNE) après un clique sur le bouton Appeler.

Un message d'erreur apparaît lors d'un clique sur le bouton appeler si l'utilisateur n'a pas rempli les paramètres.

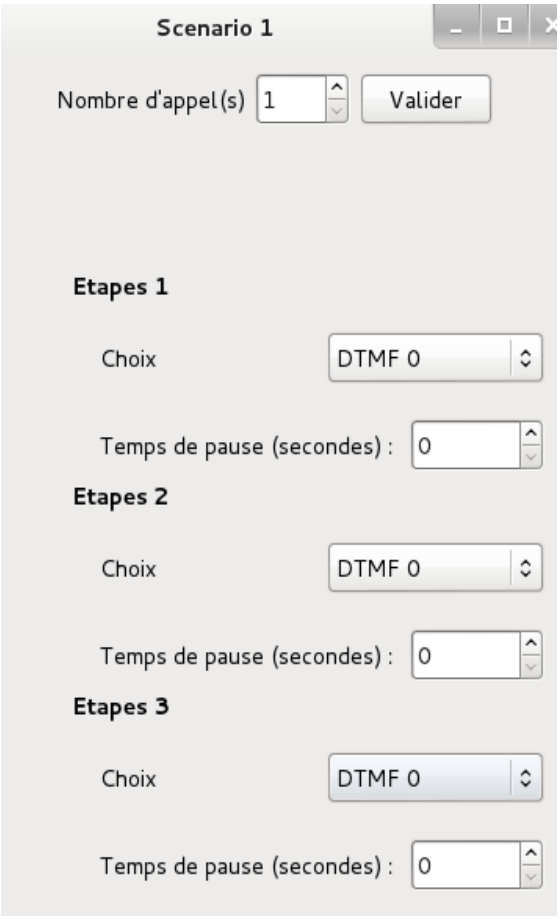


Nous allons maintenant voir le paramétrage pour un exemple de scénarios: Ici le scénario 1.

Le nombre d'étapes sera généré automatiquement en fonction du nombre d'étapes choisi.

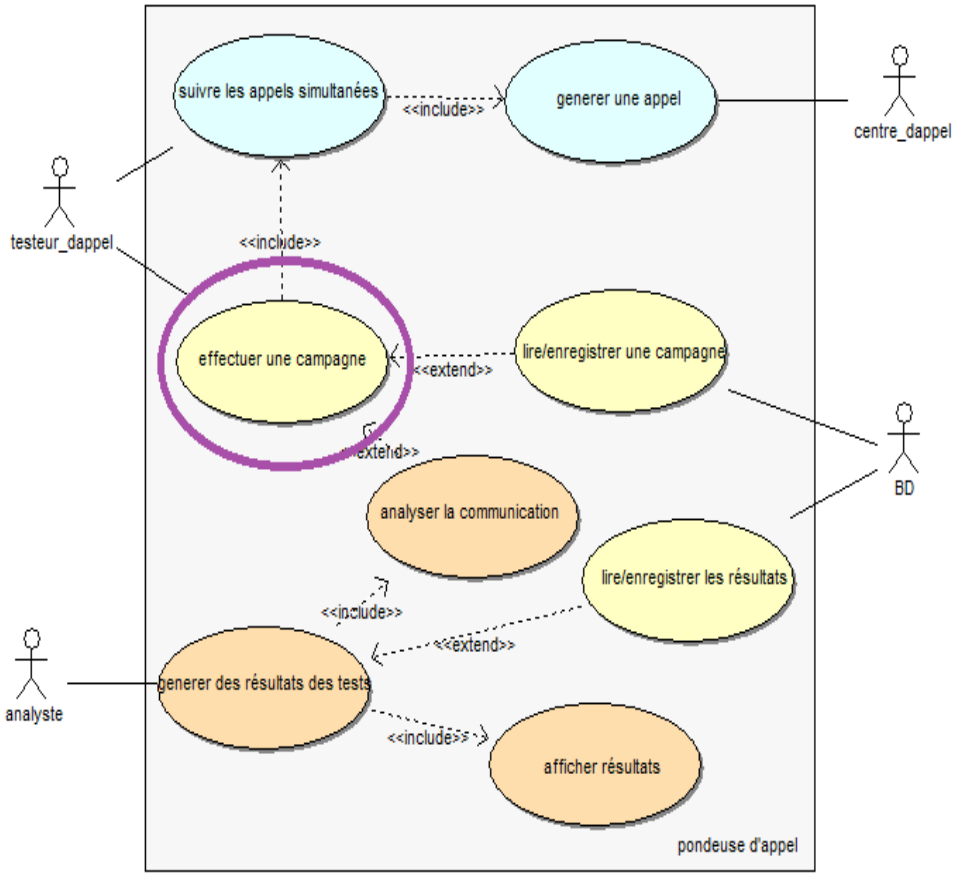
Un temps de pause est requis entre chaque étape,il peut être égal à 0,celui ci est exprimé en secondes.

Le bouton Valider valide les paramètres choisis et renvoie à l'IHM de la campagne.



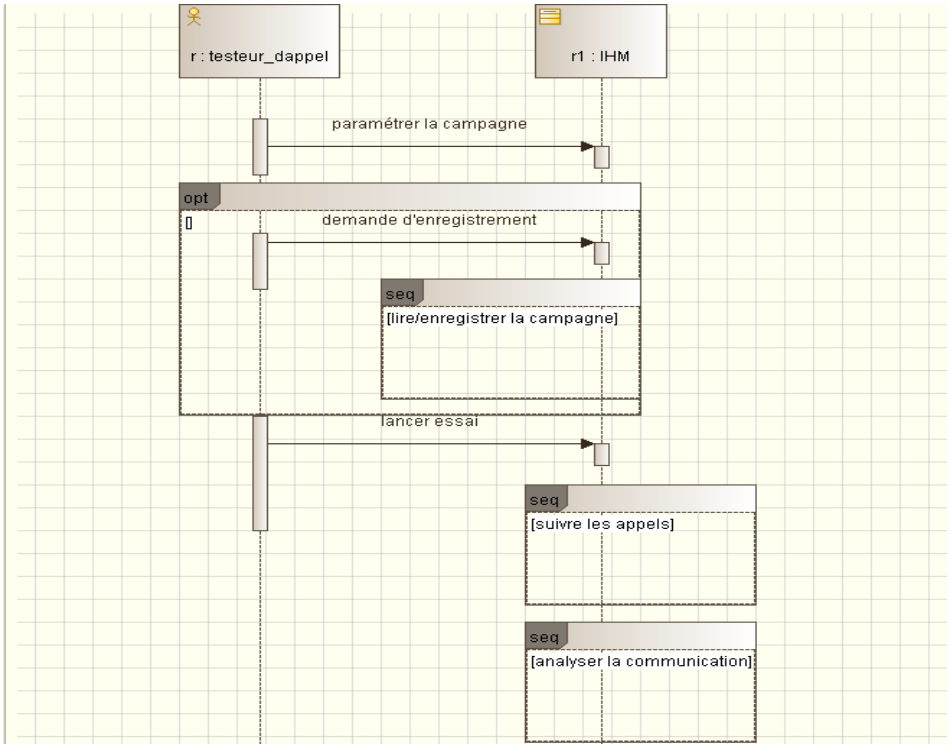
3.5 Les différents diagrammes :

Nous nous situons sur le cas d'utilisation effectuer une campagne.



Voici le diagramme de séquence du cas d'utilisation effectuer une campagne d'appels.

Lorsque l'utilisateur à entrer toutes les informations nécessaires dans l'IHM (nombre de scénarios, différentes étapes...) la campagne peut être envoyée.



L'utilisateur a la possibilité de sauvegarder une campagne ou de lire une ancienne campagne. Elle est composée des paramètres des appels effectués simultanément et des conditions de fin de campagne ainsi que des conditions de test.

Parmi les conditions de tests nous trouvons le temps de l'essai envisagé, les éléments de qualité de service qui seront définis par l'analyste et enfin les numéros d'appels.

Les conditions de fin de campagne seront définies par la pondeuse d'appels.

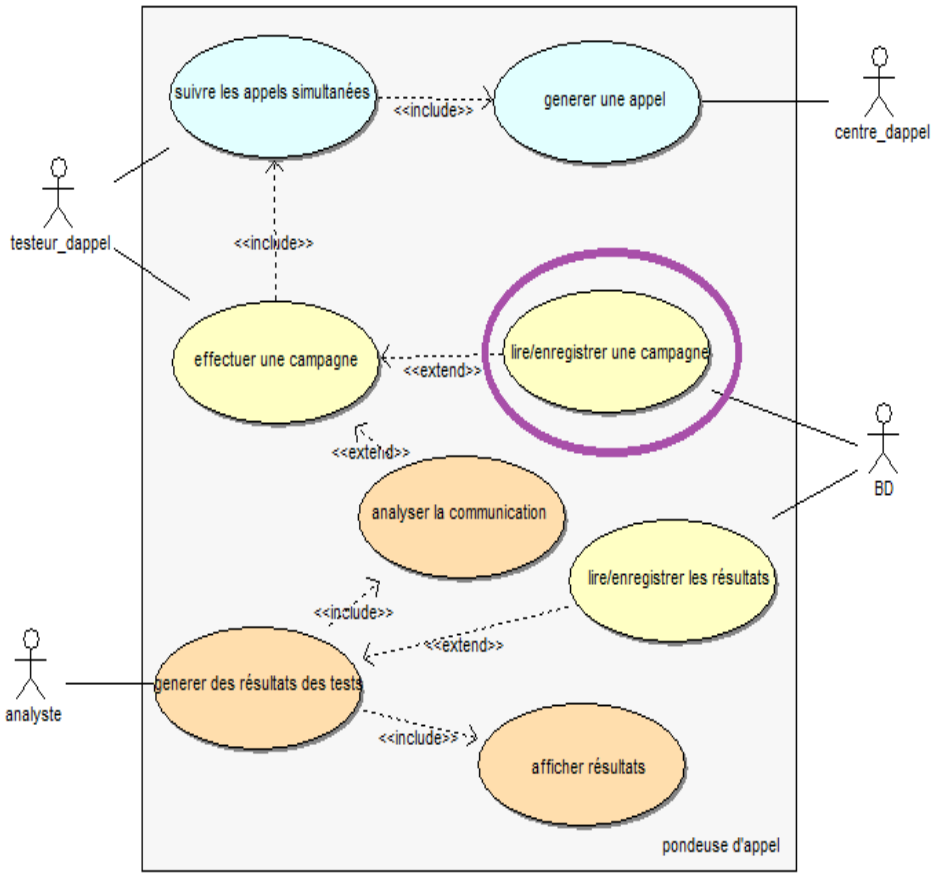
Nous avons donc 2 acteurs, qui sont :

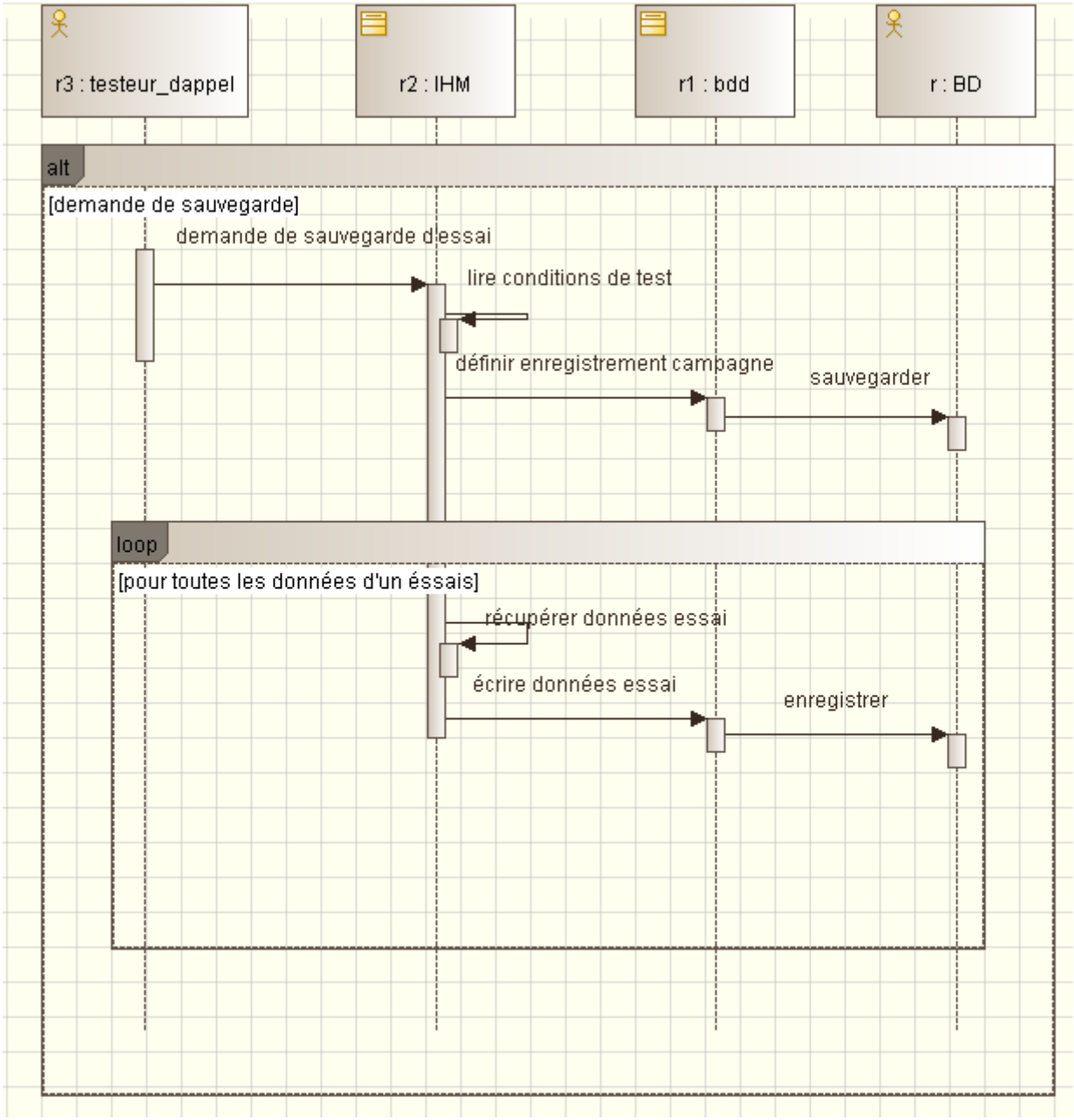
- testeur_appel : Le testeur est chargé de réaliser des tests.
- Base de données : La base de données contient toutes les informations nécessaires au bon fonctionnement du système.

Les classes :

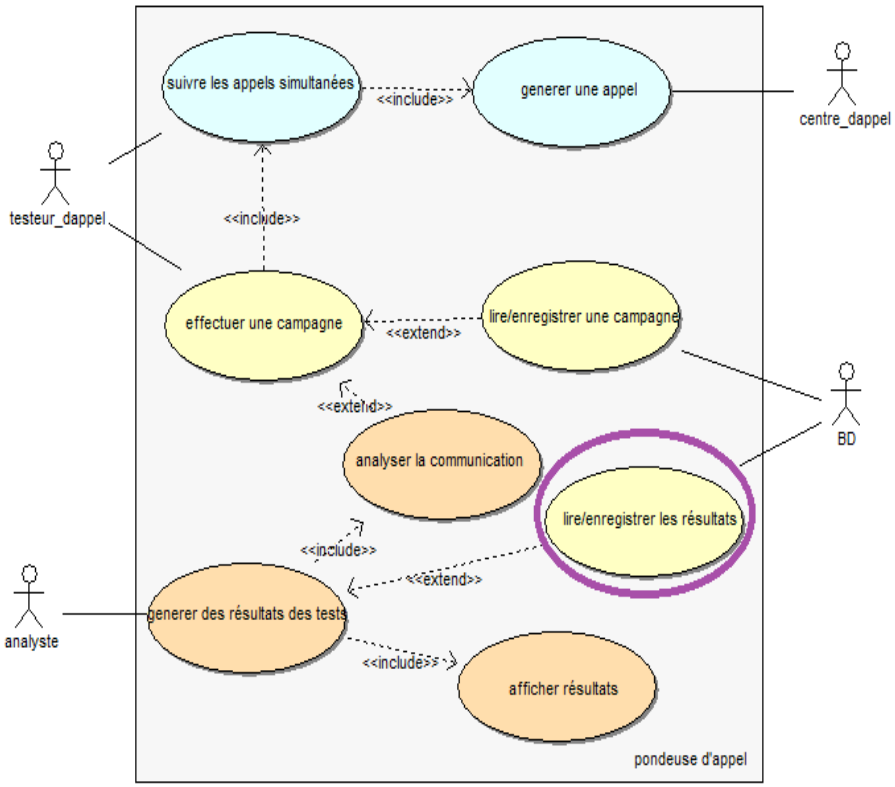
- IHM, cette classe représente les IHM, ce package va permettre la communication entre le système et le testeur.
- Bdd, cette classe va permettre l'accès aux informations de la base de données, aussi bien pour stocker que lire les données.

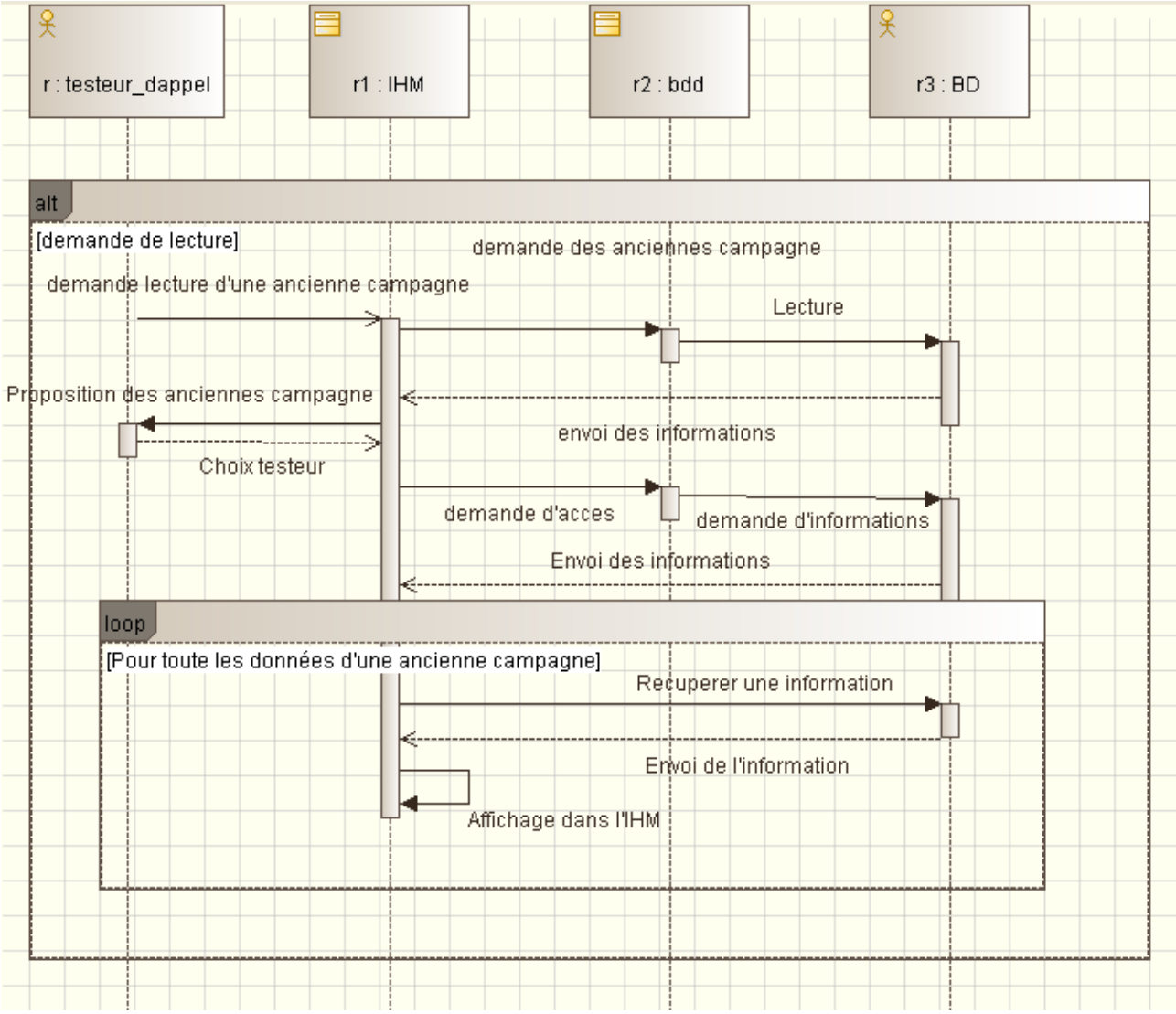
Voici le diagramme de séquence du cas d'utilisation, enregistrer une campagne :


suivre les appels simultanées
<<include>>
generer une appel
centre_dappel
testeur_dappel
<<include>>
effectuer une campagne
<<extend>>
lire/enregistrer une campagne
<<extend>>
BD
analyser la communication
lire/enregistrer les résultats
<<include>>
<<extend>>
analyste
generer des résultats des tests
<<include>>
afficher résultats
pondeuse d'appel


r3 : testeur_dappel
r2 : IHM
r1 : bdd
r : BD
alt
[demande de sauvegarde]
demande de sauvegarde d'essai
lire conditions de test
définir enregistrement campagne
sauvegarder
loop
[pour toutes les données d'un éssais]
récupérer données essai
écrire données essai
enregistrer

Voici le diagramme de séquence du cas d'utilisation, lire une campagne :




r : testeur_dappel
r1 : IHM
r2 : bdd
r3 : BD
alt
[demande de lecture]
demande des anciennes campagne
demande lecture d'une ancienne campagne
Lecture
Proposition des anciennes campagne
envoi des informations
Choix testeur
demande d'acces
demande d'informations
Envoi des informations
loop
[Pour toute les données d'une ancienne campagne]
Recuperer une information
Envoi de l'information
Affichage dans l'IHM

La pondeuse d'appels génère des appels vers le centre d'appel qui va traiter ces appels. Nous allons maintenant nous placer du côté de la pondeuse d'appels.

La pondeuse d'appels a besoin d'un poste sur lequel se trouvera l'application pour envoyer les threads (géré par l 'étudiant 3) à la Trixbox .

Le PC de contrôle sert à la configuration du PABX ainsi que l'utilisation du logiciel CCsupervision .

La Trixbox communique avec le l'Alcatel OMNI-PCX ENTERPRISE via un faisceau IP (Trunk).

Les appels sont traités par le centre d'appels.

3.5.1 Le diagramme de classe

Principales classes :

Les numéros seront définis comme des chaînes de caractères (string).

L'IHM permettra aussi a l'utilisateur de sauvegarder la campagne ou d'en charger une ancienne, nous pouvons donc définir plusieurs méthodes :

- sauvegarderCampagne() ; Pour sauvegarder une campagne
- chargerAncienneCampagne() ; Pour charger un ancienne campagne
- supprimerCampagne() ; Pour supprimer une campagne qui a été enregistrée
- envoyer() ; Pour démarrer une nouvelle campagne

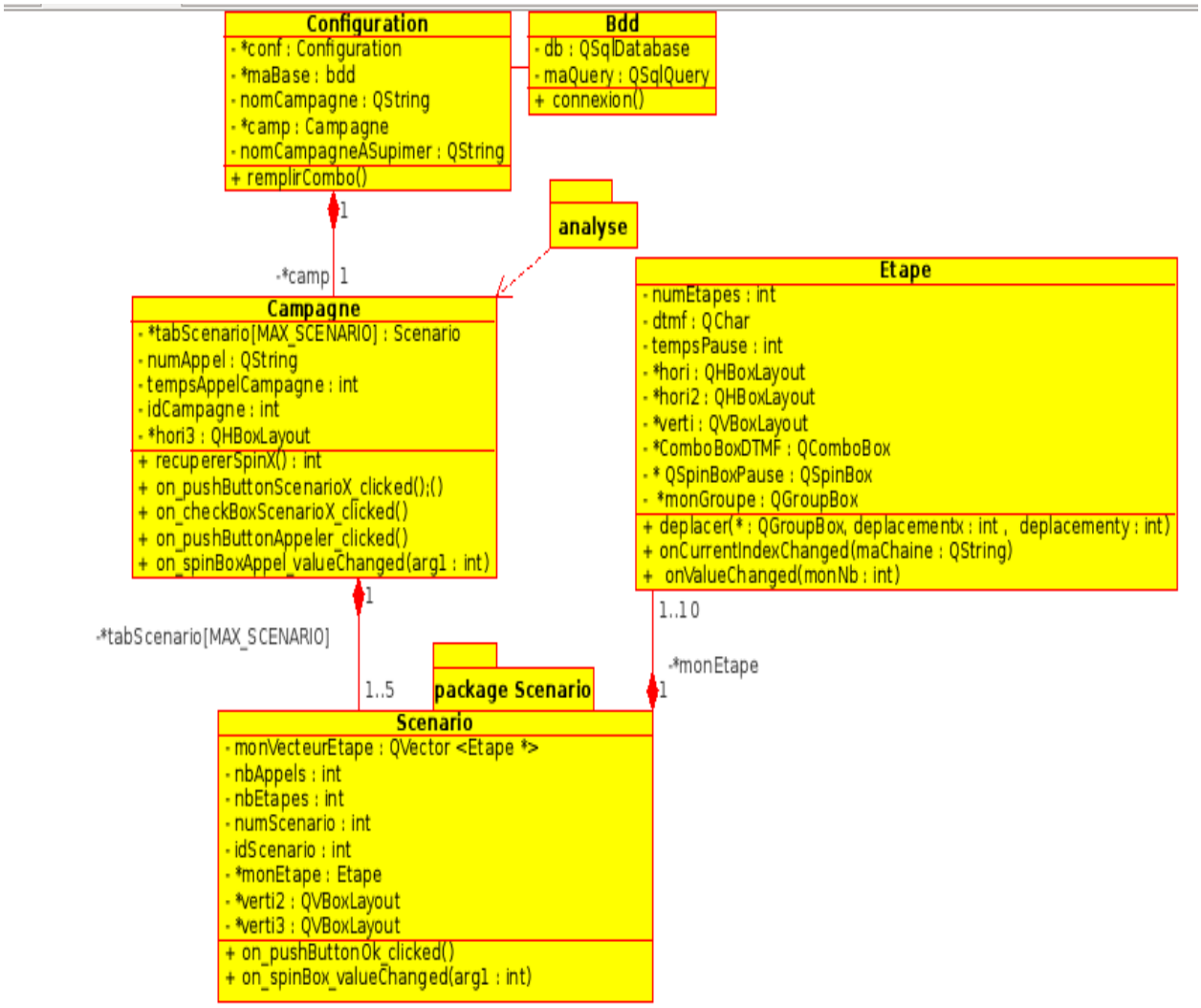
La première classe nécessaire est la classe Configuration, cette classe permettra de choisir le nom de la campagne. C'est donc la configuration de la campagne.

La deuxième classe nécessaire est la classe Campagne, cette classe regroupera tout les besoins d'une campagne qui sont :

- le numéro de l'appel
- la durée maximum d'appel

Références de fréquence :

- Les 2 packages sont le package Analyse géré par Florian DAUVERGNE(étudiant 3) et le package Scenario géré par Clément CHRETIEN(étudiant 2).



3.1- Présentation du système signal/slot

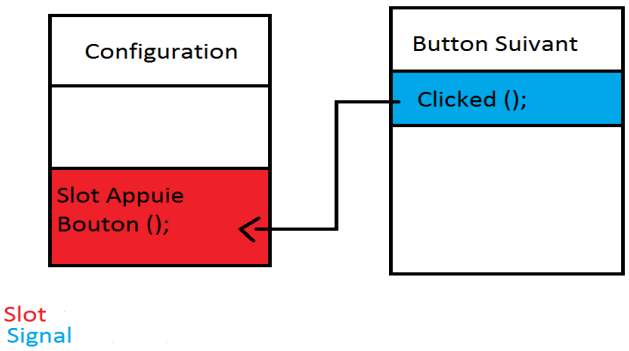
Sur l'IHM, une liaison entre les différents éléments du formulaire permet de faciliter l'utilisation pour l'administrateur. Ceci, principalement grâce au système signal/slot de Qt mis en place notamment grâce à l'outil qtCreator.

Un signal est un message envoyé par un widget lorsqu'un événement se produit. (Exemple : un clic sur un bouton ou le changement d'une valeur dans une QSpinBox).

Un slot est une fonction appelée lorsqu'un événement s'est produit. Il est appelé par le signal.
Un slot est une méthode d'une classe, dans notre application il s'agit de méthode des différentes classes qui possèdent une IHM.
(Exemple : le slot quit() appartenant à la classe Qwidget de base qui fermera la fenêtre)

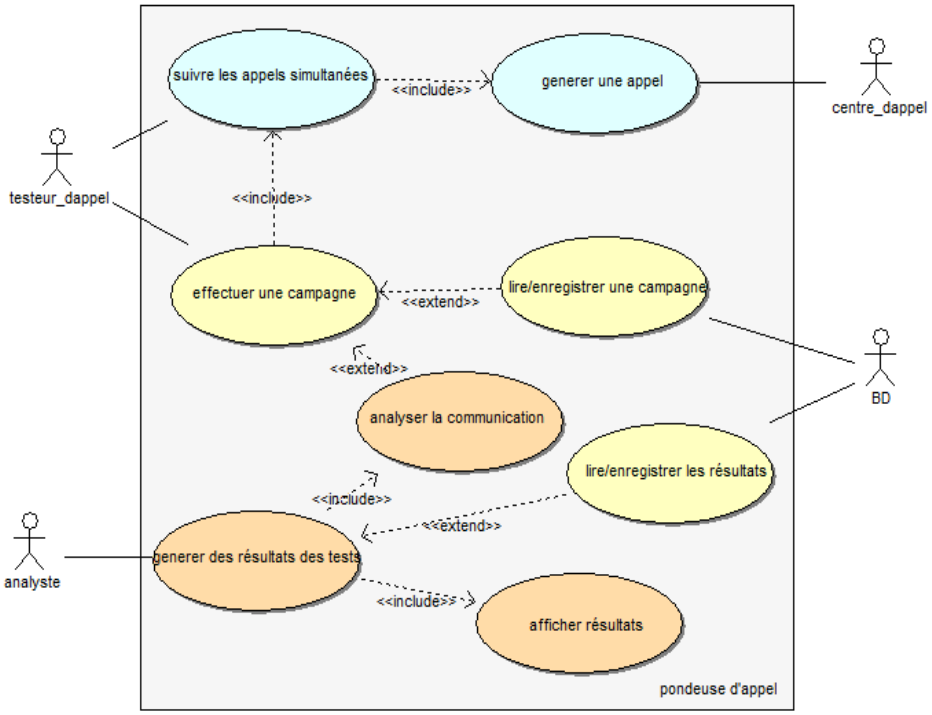
Qt offre la possibilité de coder le contenu de nouveau slot. Cette association nous permettra donc d'appeler des méthodes pour modifier l'IHM lors d'évènements précis afin de guider l'utilisateur.

Schéma de représentation du système Signal/Slot:



Dans le schéma, lorsque le button1 est cliqué, le slot de l'objet de la classe Configuration nommé
« on_pushButtonSuivantConf_clicked(); » est appelé.

3.6 Les interactions :



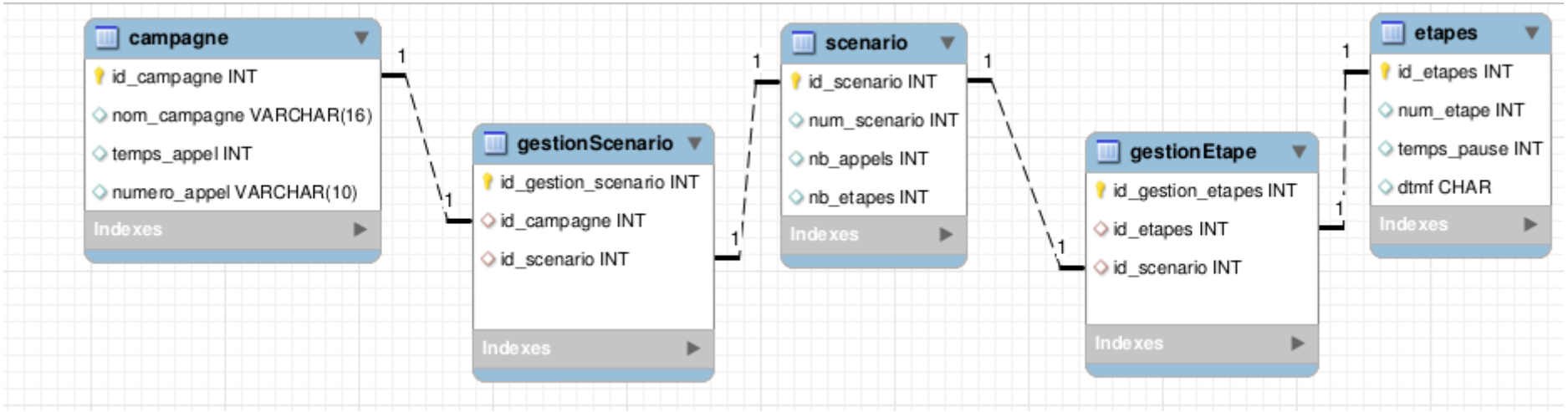
Je vais être en relation avec 2 acteurs : La base de données et le testeur d'appels. Je vais être aussi en relation avec Clément CHRETIEN(étudiant 2) pour lui fournir les paramètres dont ils a besoin pour générer les N.appels, Florian DAUVERGNE(étudiant 3) lui, va devoir récupérer les informations des paramètres de la campagne et analyser les appels.

3.7 Les tables de la base de données

Les tables ont été choisies en fonction des besoins du projet ainsi que par déduction en fonctions des différents diagrammes et des IHM.

Les tables suivantes sont mes tables, je vais les renseigner et les consulter.

- campagne(elle stocke les informations liées aux différentes campagnes)
- gestionScenario (elle fait le lien entre la table campagne et scénario, une campagne pour un ou plusieurs scénarios)
- scénario (elle stocke les informations liées aux différents scénarios)
- gestionEtape(elle fait le lien entre la table scenario et étapes, un scénario pour une ou plusieurs étapes)
- étapes (stocke les informations liées aux différentes étapes)



3.8 Travail futur :

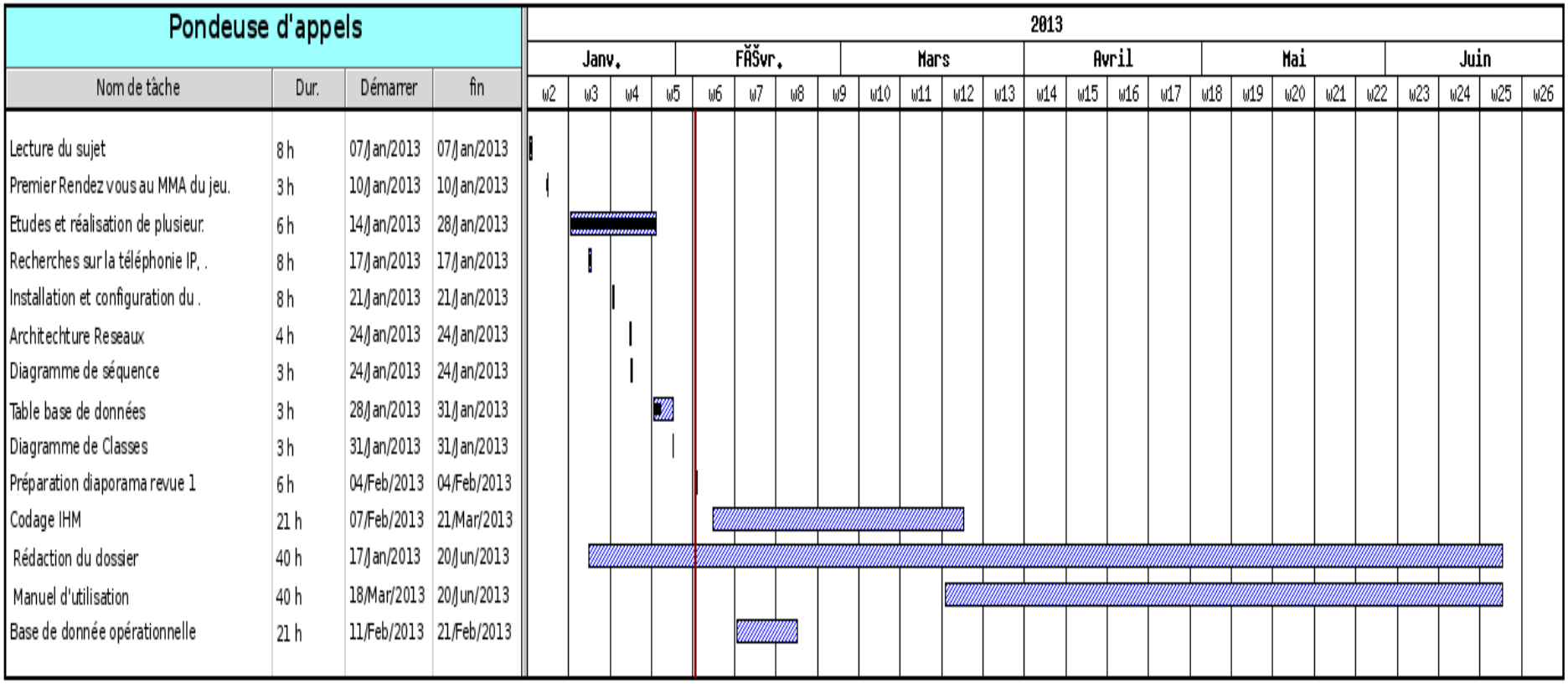
Le plus gros travail pour ma part va être le codage des IHM, qui sera réalisé sous QtCreator.

La base de données sera à mettre en place aussi, cela s'effectuera sur le serveur Trixbox.

Une partie qui va prendre du temps sera la phase de mise en commun, car il faudra que tout notre travail s'articule pour finaliser le projet.

Une fois ce travail achevé, il faudra rédiger le manuel d'utilisation.

3.9 Diagramme de gantt


Pondeuse d'appels
2013
Janv.
FÃŠvr.
Mars
Avril
Mai
Juin
w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12 w13 w14 w15 w16 w17 w18 w19 w20 w21 w22 w23 w24 w25 w26
Nom de tâche | Dur. | Démarrer | fin
Lecture du sujet | 8 h | 07/Jan/2013 | 07/Jan/2013
Premier Rendez vous au MMA du jeu. | 3 h | 10/Jan/2013 | 10/Jan/2013
Etudes et réalisation de plusieur. | 6 h | 14/Jan/2013 | 28/Jan/2013
Recherches sur la téléphonie IP, . | 8 h | 17/Jan/2013 | 17/Jan/2013
Installation et configuration du . | 8 h | 21/Jan/2013 | 21/Jan/2013
Architechture Reseaux | 4 h | 24/Jan/2013 | 24/Jan/2013
Diagramme de séquence | 3 h | 24/Jan/2013 | 24/Jan/2013
Table base de données | 3 h | 28/Jan/2013 | 31/Jan/2013
Diagramme de Classes | 3 h | 31/Jan/2013 | 31/Jan/2013
Préparation diaporama revue 1 | 6 h | 04/Feb/2013 | 04/Feb/2013
Codage IHM | 21 h | 07/Feb/2013 | 21/Mar/2013
Rédaction du dossier | 40 h | 17/Jan/2013 | 20/Jun/2013
Manuel d'utilisation | 40 h | 18/Mar/2013 | 20/Jun/2013
Base de donnée opérationnelle | 21 h | 11/Feb/2013 | 21/Feb/2013

3.10 Conclusion :

Cette première partie de projet se déroule bien, il y a une bonne ambiance de travail, les choix conviennent à l'équipe.

Ce projet de téléphonie est nouveau pour nous, il a donc fallu faire beaucoup de recherches afin de travailler convenablement.

Pour la suite :
- un travail de codage de longue haleine
-Une dépendance de Quentin PRIEUR (étudiant 4), car si le centre d'appels n'est pas mis en place les appels ne pourront pas être passés.

4. Conception préliminaire : Partie Chrétien Clément (E2)

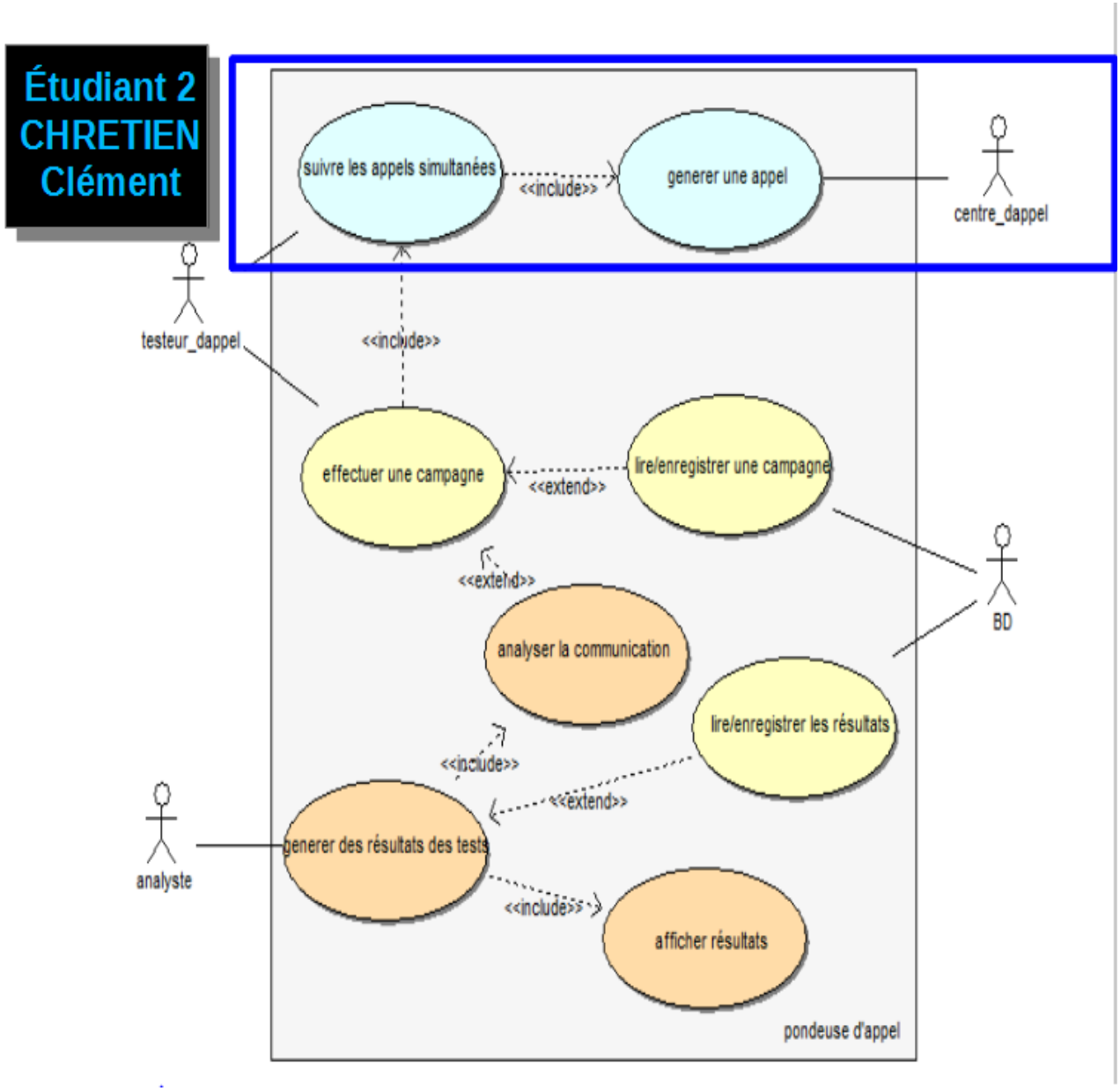
Contexte :

Les MMA ,groupe d'assurances mutuelles souhaitent tester ses infrastructures téléphoniques administratives ou CCD(Contact Center Distribution, téléphonie de centre d'appel) afin de tester leur robustesse et leurs limites.C'est pourquoi l'entreprise nous a confié la réalisation d'une pondeuse d'appels permettant de générer un certain nombre d'appels(Campagne d'appels) dans le but de récupérer les informations liées à cette campagne et les retranscrire sous forme de statistiques.

Pour aboutir à ce projet, nous l'avons divisé en quatre grandes parties. Étant l'étudiant 2, les tâches principales que je dois exécutées sont :

- générer un appel
- suivre les appels simultanés

Voici la place des taches qui me concerne dans le cas d'utilisation général:



Liste des tâches :

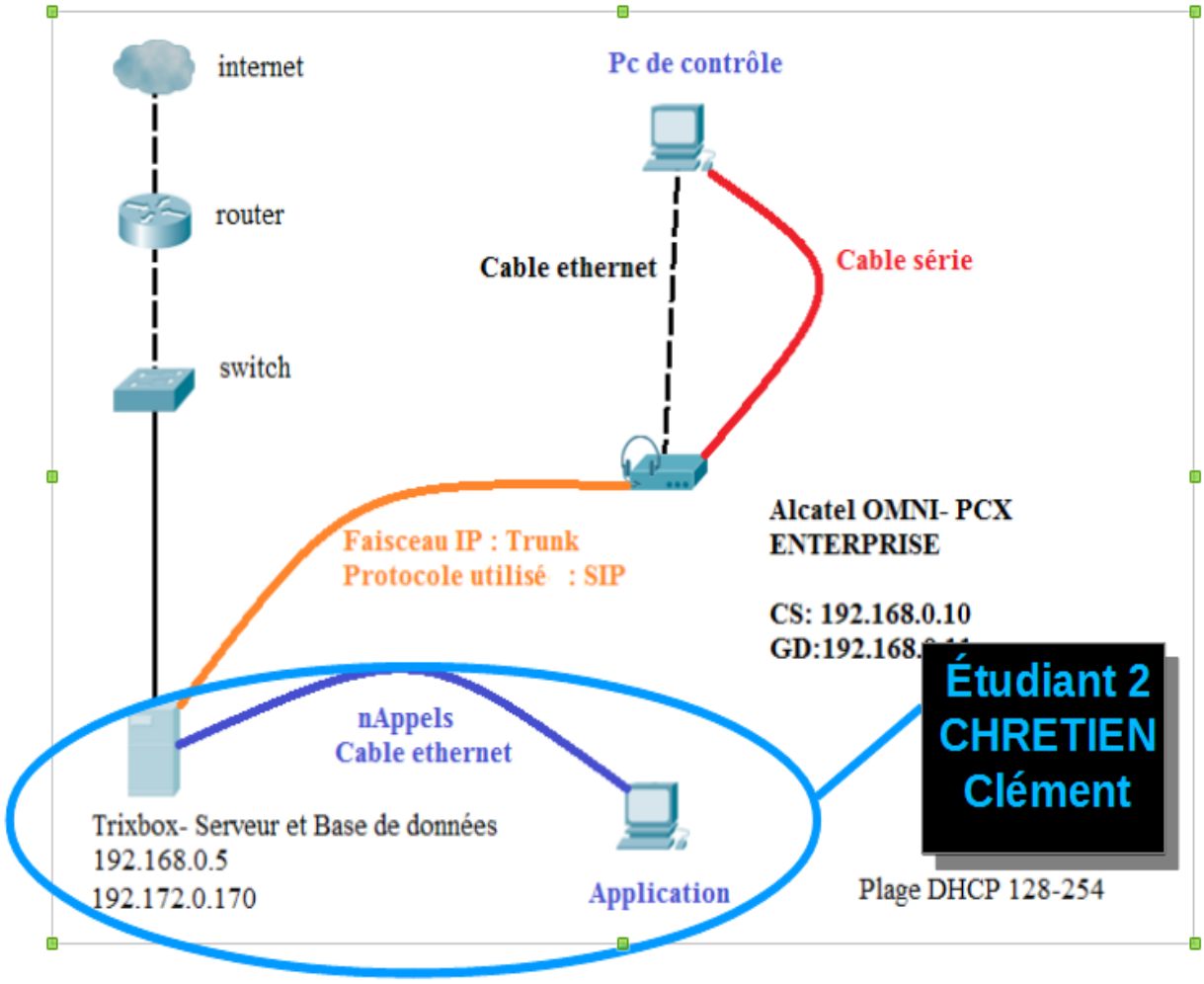
- Prise de connaissance du sujet.
- Création d'un planning prévisionnel
- documentation sur la téléphonie
- décrire les étapes clés d'un appel
- Choix codage, EDI
- trouver solution pour générer un appel à l'aide de la trixbox
- trouver solution pour générer des appels simultanés
- mis en place de la pondeuse d'appels
- définition d'un premier diagramme de classe

- Visite de l'entreprise et mise au point avec les interlocuteurs.
- Préparation à la revue 1 et création du dossier.
- Codage de l'application d'administration du site
- Test unitaire

4.1 Étude préliminaire:Suivre les Appels simultanés

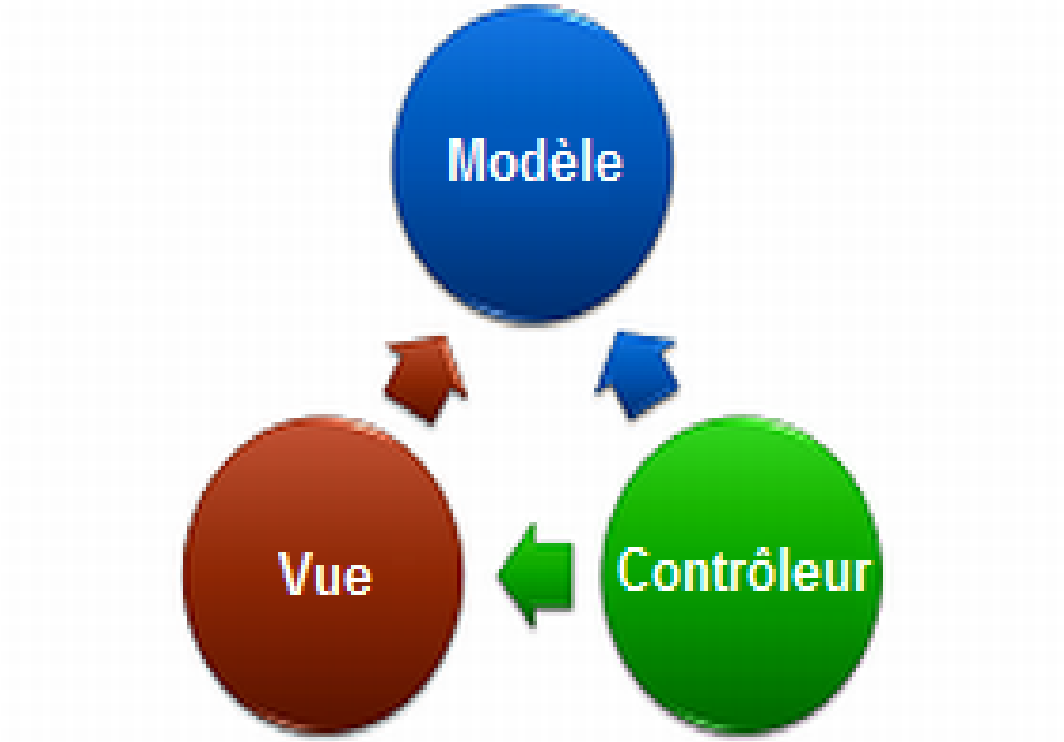
Lorsque l'utilisateur souhaitera effectuer une campagne d'appel à l'aide de l'application, il devra être connecté à la trixbox sur le réseau local ou sur celui de la trixbox.

La pondeuse d'appels devra alors générer simultanément le nombre d'appels définis dans l'application pendant le temps imparti ou jusqu'a saturation du centre d'appels. Ces appels devront avoir une réponse dans un temps compatible avec les valeurs du centre d'appels, mais aussi une qualité d'écoute acceptable. À la fin de cette campagne, le nombre d'appels simultané effectué devra être enregistré.



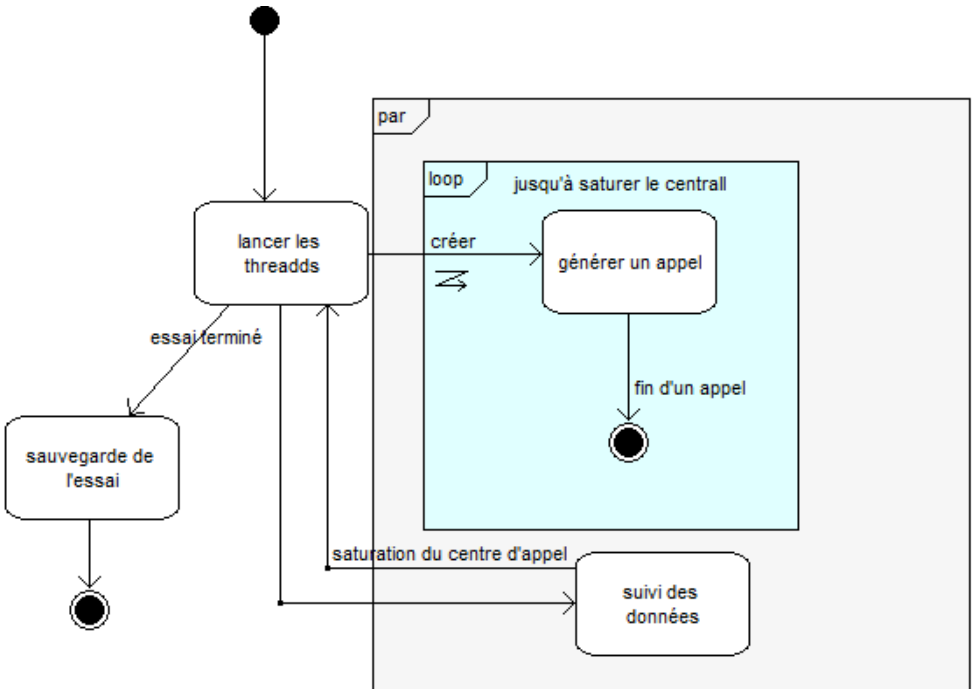
L'application est décomposée entre les étudiants et respecte le modèle MVC (Model View Controller).

Mon travail est celui du contrôleur, je synchronise les données entre le modèle et la vue, je m'occupe de toute la partie codage de l'application de la pondeuse.



4.1.1 Diagramme d'état transition d'un essai

L'application doit générer simultanément un ensemble d'appels. Pour ce faire l'utilisation de threads est indispensable d'une part pour que les appels se déroulent de façon simultanée, mais d'autre part pour la récupération des informations liées à un appel. Les threads permettront de différencier chaque appel de façon unique. Les threads seront codés sous QT creator en c++ grâce à la classe QThread. Nous avons choisi QT creator pour sa facilité d'utilisation , le fait qu'on a déjà travaillés avec QT creator, ce logiciel fonctionne sur la plupart des systèmes d'exploitation.



Ce diagramme d'état transition montre le déroulement d'un essai. Après que l'utilisateur est démarré l'application le programme lance les threads et dans chaque thread on génère un appel si le nombre d'appels définis est effectué on sauvegarde toutes les données liées aux appels et on stop les threads. S'il ya saturation du centre d'appel avant la fin prévue de la campagne la totalité de l'application est stoppée et on effectue la sauvegarde des informations (ici échec de la campagne).

4.1.2 Les données de suivi

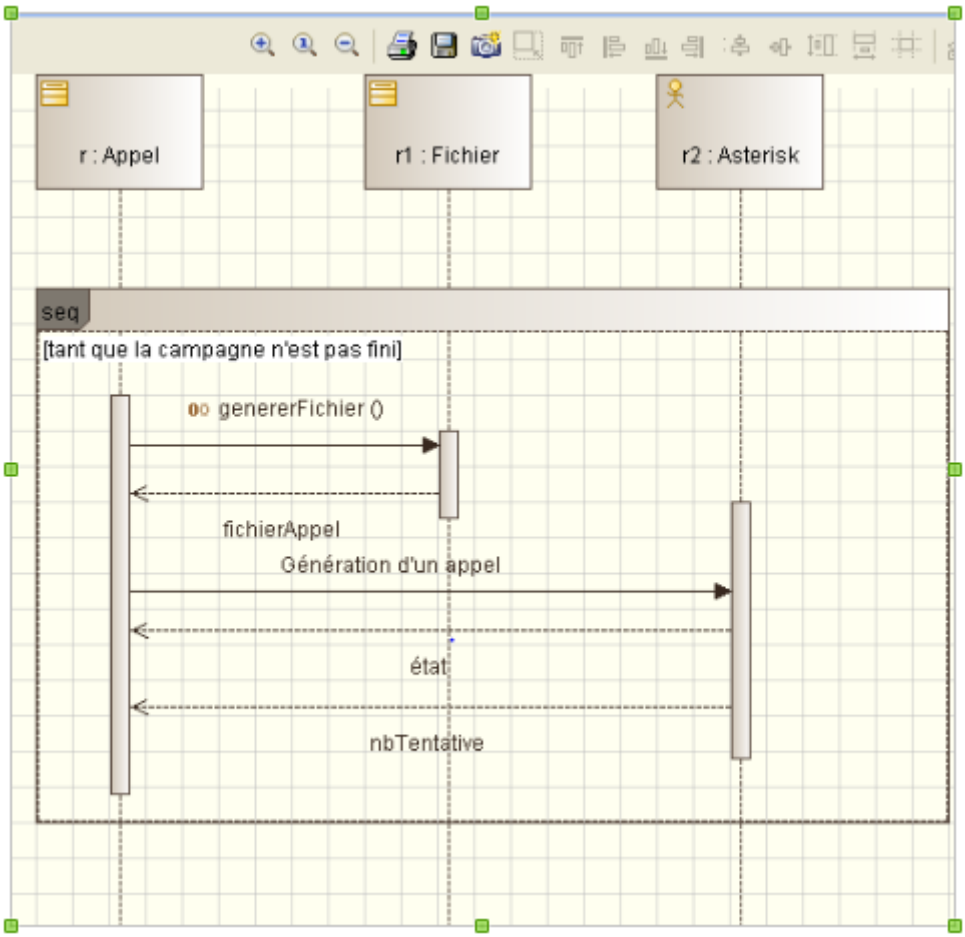
Après la fin de la campagne ou jusqu'à saturation du centre d'appel, les données de l'essai doivent être enregistré. Voici les différentes données de suivi :

- le nombre d'appels simultanés effectuer pendant la campagne
- le numéro appelé
- la date de début de l'essai (date et heure)

- la date de fin de l'essai (date et heure)
- le nombre d'appels avorté
- le nombre d'appels réussi
- le nombre de tentatives de chaque appel

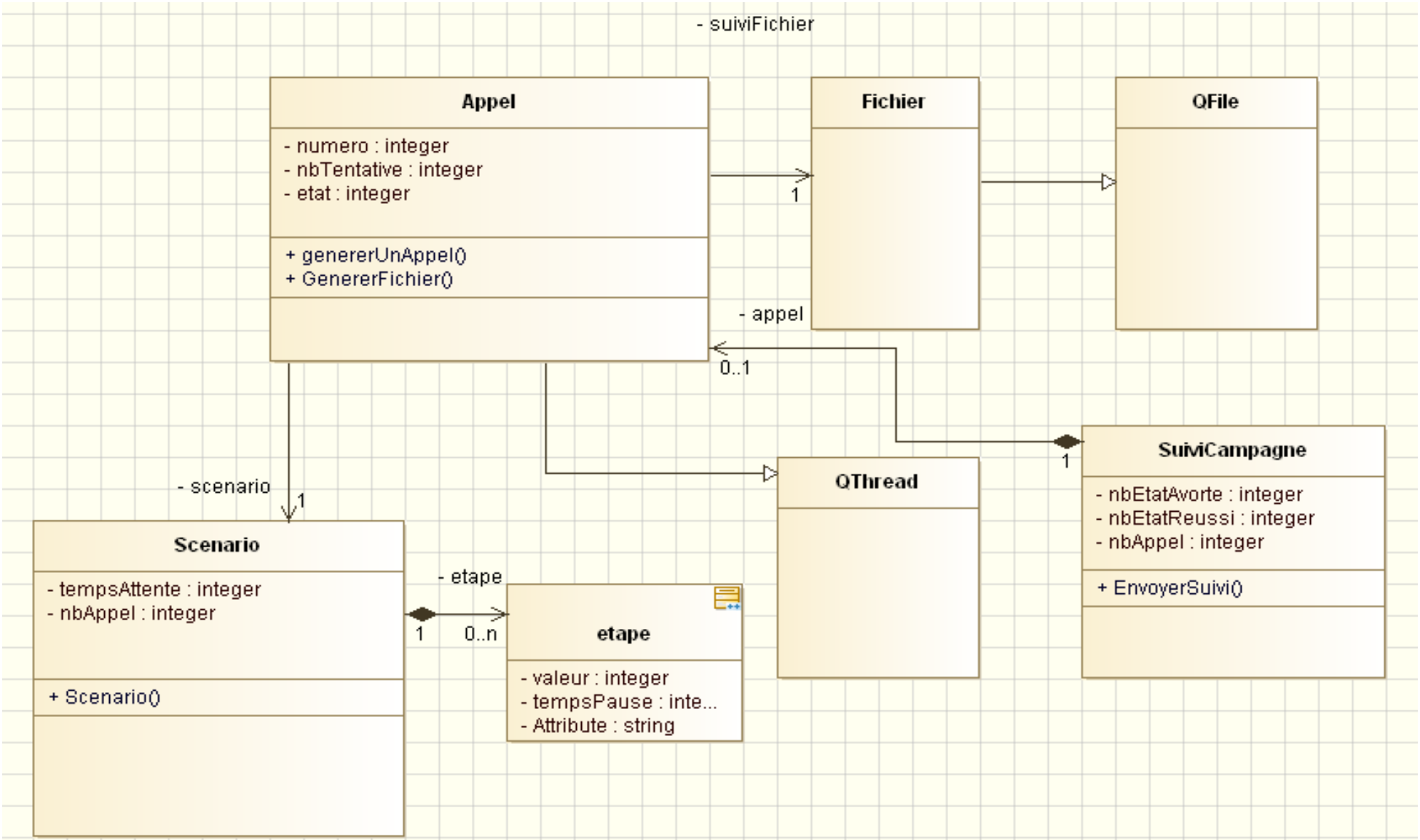
Avant de passer un appel on doit vérifier que la ligne est disponible, on effectue plusieurs tentatives si la ligne n'est pas disponible il nous faut donc savoir combien de tentatives a-t-il fallu avant que l'appel ne soit émit.

4.1.3 Diagramme de séquence pour un appel :



4.1.4 Les principales classes de l'application

D'après le diagramme de séquence, on en déduit les classes suivantes :


- suiviFichier
Appel
- numero : integer
- nbTentative : integer
- etat : integer
+ genererUnAppel()
+ GenererFichier()
Fichier
QFile
1
- appel
0..1
1
SuiviCampagne
- nbEtatAvorte : integer
- nbEtatReussi : integer
- nbAppel : integer
+ EnvoyerSuivi()
QThread
- scenario
1
Scenario
- tempsAttente : integer
- nbAppel : integer
+ Scenario()
- etape
1
0..n
etape
- valeur : integer
- tempsPause : inte...
- Attribute : string

Ce diagramme de classe représente les principales classes de l'application,celles-ci sont susceptibles d'être modifiées au cours du projet.

La classe QThread :

- la classe QThread est une classe de , elle fournit des threads indépendants de la plateforme.
- Un QThread représente un thread de contrôle séparé dans le programme ; il partage des données avec tous les autres threads dans le processus, mais s'exécute de manière indépendante, comme un programme séparé sur un système d'exploitation multitâches. Au lieu de démarrer dans **main()**, un QThread débute son exécution dans **run()**.

La classe QFile :

- la classe QFile est une classe de QT qui fournit une interface pour la lecture et l'écriture de fichiers.
- QFile est un dispositif d'entré / sortie pour lire et écrire des fichiers texte et binaire et des ressources .

La classe Fichier :

- la classe Fichier hérite de QFile ceci est fait pour pouvoir profiter des méthodes et des attributs de QFile tout en rajoutant d'autres méthodes ou attributs si besoin est.

La classe Appel :

- la classe appel hérite de QThread pour que chaque appel puisse être exécutés en même temps. Elle a pour but de générer les appels de la pondeuse.
- Le numéro est le numéro de téléphone qui doit être appelé. Il est de type entier.
- NbTentative :est le nombre de tentatives pour un appel si la ligne est occupée au-delà, l'état de l'appel est considéré, comme « avorté ». Il est de type entier.
- État : est l'état d'un appel il varie en fonction de l'avancement de l'appel ou comme il est dit plus au-dessus en cas d'échec. Les valeurs de l'état peuvent être : « réussi », « avorté », « en attente », « en cour ». Il est de type String.
- Appel():c'est le constructeur de la classe Appel.
- genererFichier():cette méthode va générer un fichier pour générer un appel, ce principe sera expliqué un peu plus loin dans le dossier.
- genererUnAppel() :cette méthode permet de générer un appel à l'aide du fichier créer par la méthode genererFichier().

La classe SuiviCampagne :

- La classe SuiviCampagne permet de stocker les informations de la campagne et

des appels passés durant cette campagne afin de pouvoir par la suite les stocker dans une base de données.

- nbEtatAvorte : est un attribut de la classe suivi, il est de type entier, il permet de compter le nombre d'appels qui n'ont pas pu être passés, car la ligne n'était pas disponible.
- nbEtatReussi : est un attribut de la classe suivi, il est de type entier, il permet de compter le nombre d'appels qui ont été correctement passés c'est-à-dire que toutes les étapes liées à l'appel se sont déroulées correctement .
- nbAppel :est un attribut de la classe suivi de type entier il compte le nombre d'appels simultanés passer durant la campagne qu'il soit ou non réussi (état « réussi » ou « avorte »).

La classe Scenario :

- la classe Scénario permet de savoir quels scénarios doivent suivent les appels durant la campagne(envoi de DTMFs , Pause...).
- tempsAttente :c'est un attribut de la classe Scenario de type entier qui donne le temps d'attente entre deux étapes du scénario .
- nom :C'est un attribut de la classe Scenario de type chaîne de caractères, c'est le nom du scénario.
- nbAppel:c'est un attribut de la classe Scenario de type entier qui donne le nombre d'appels qui doivent être liés à ce scénario.
- La classe Scenario est composée d'étapes, une structure qui contient le nom de l'étape, sa valeur ainsi que le temps de pause accordé après cette étape.

4.2 Étude préliminaire:Générer un appel

Après la création des threads il faudra à l'intérieur de chaque thread générer un appel suivant le scénario défini dans l'IHM conçu par l'étudiant 1 : Mathias PLATEAU.

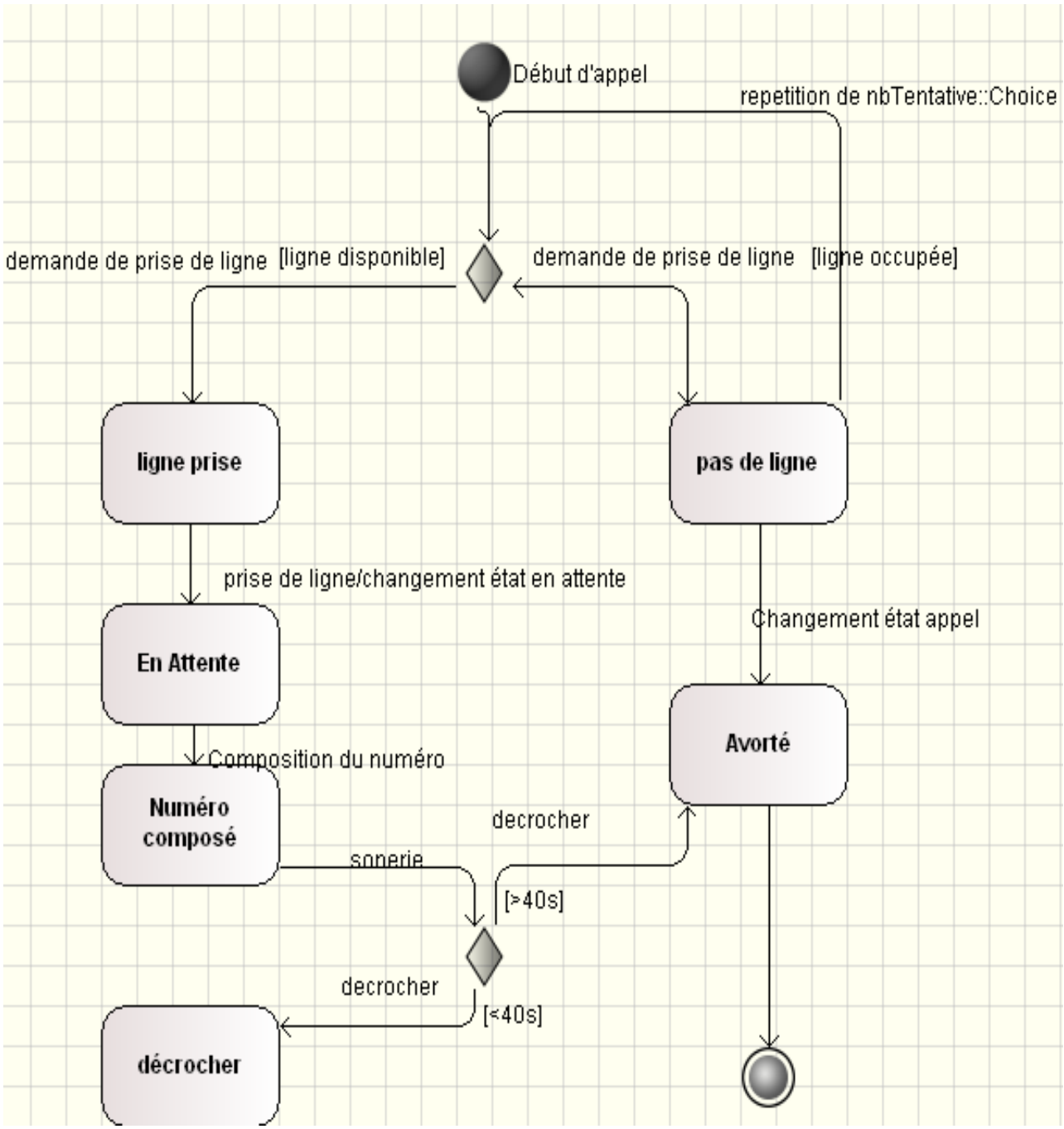
Le scénario se déroule de façon identique seule les étapes qui sont définies dans l'IHM vont modifiés ce scénario.

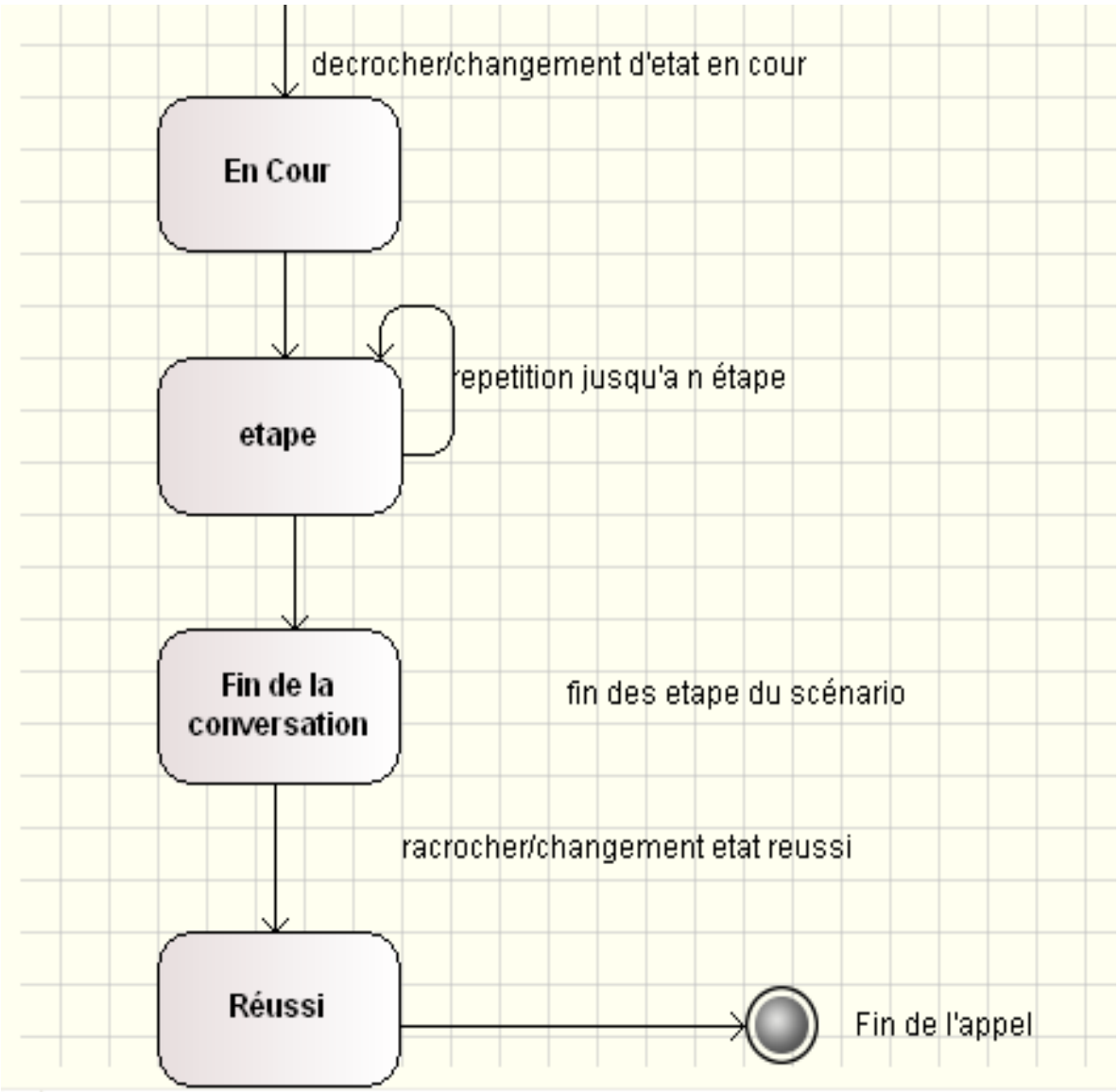
Voici l'algorithme d'un appel :

- SI la ligne est disponible
  - prise de ligne
  - composition du numéro
  - etat= « en attente »
  - décrocher
  - etat= « en cours »
  - POUR indice allant de 0 jusqu'à nbEtapes

        - etape indice
    - FIN POUR
    - raccrocher
    - etat= « réussi »
- FIN SI
- SI ligne est indisponible
    - etat= « avorté »
    - raccrocher
- FIN SI

4.2.1 Diagramme d'état transition d'un appel





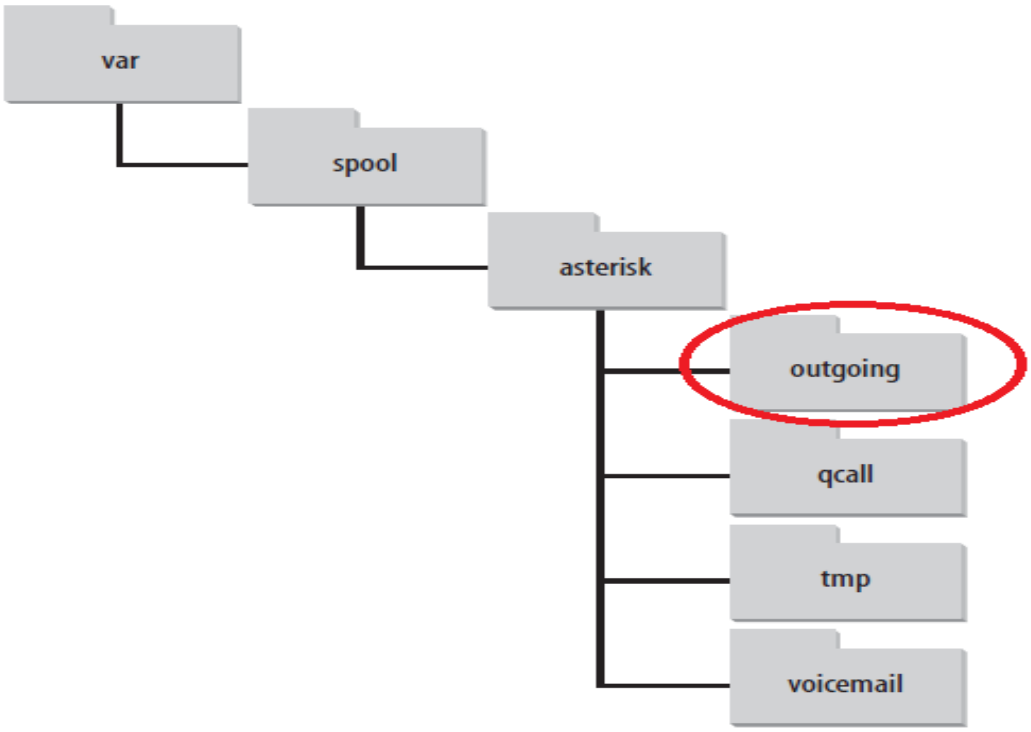
Ce diagramme d'état permet de représenter les changements d'état durant un appel généré.

4.2.2 Solution trouvée pour la génération d'appel à partir du serveur Asterisk

Les appels doivent être générés à l'aide d'un serveur téléphonique (trixbox) commandé par un ordinateur client sur lequel il y aura l'application.

Solution trouvée :

Après des recherches une solution a été trouvée elle se nomme **AutoCall.** Cette méthode consiste à générer des appels automatiquement à l'aide de fichiers que l'ont placent dans un répertoire d'Asterisk.



Dans un premier temps ,il faut configurer un scénario appelé **context** dans le fichier **extensions.conf** situé dans le répertoire, **etc/asterisk** sinon il prendra le scénario par défaut c'est-à-dire que l'appel sera émis et il lancera le fichier audio goodbye.ulaw puis l'appel sera terminer.

Ensuite, il faut éditer un fichier avec une extension .call c'est le fichier qui sera à placer dans le répertoire **outgoing**. Ce fichier prend plusieurs paramètres pour les tests préliminaires nous avons utilisé un fichier simple qui lit juste un fichier audio pour simuler une conversation.

Voici les paramètres que peut prendre le fichier .call :

- Channel : c'est le numéro qui sera appelé.
- WaitTimes:c'est le temps d'attente pour un appel sans réponse avant que l'appel soit considéré comme « avorté ».
- RetryTimes:c'est le temps d'attente entre chaque tentative.

- MaxRetries:c'est le nombre de tentatives si la ligne n'est pas disponible.
- Context:c'est le scénario que l'appel devra suivre.
- Extension:c'est là ou l'appel devra se connecter.
- Priority:c'est la priorité d'un appel par rapport aux autres.

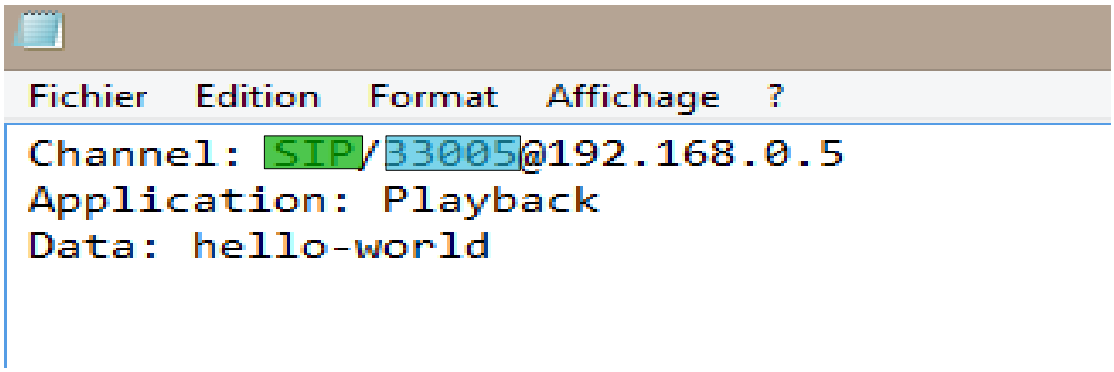
Alternativement, on peut spécifier une seule fonction et lui passer un paramètre :

- Application : ici, on place le nom de la fonction comme dans le test on peut placer:Playback
- Data :ici, on place le paramètre de la fonction juste au-dessus par exemple « Hello-world », qui est un son présent par défaut sur Asterisk.

Ensuite on place les informations de l'appelant :

- CallerID:c'est le nom de l'appelant qui sera affiché sur le téléphone du destinataire.

4.2.3 Exemple de fichier :



Voici le fichier utilisé pendant le test, ce fichier va appeler à l'aide du protocole SIP le numéro de téléphone 33005 et va jouer en arrière-plan grâce à l'application Playback le fichier audio hello-world.ulaw format issue de la compression d'un fichier audio avec le codec ulaw très gourmand en bande passante, mais avec une compression de donné assez faible pour garder le son intact.

Après avoir générer le fichier il faut le placer dans le répertoire /var/spool/asterisk/outgoing à l'aide de la commande :

```
mv nomDuFichier.call  /var/spool/asterisk/outgoing/
```

ou

```
cp nomDuFichier.call  /var/spool/asterisk/outgoing/
```

suite a ceci asterisk génère automatiquement l'appel correspondant au fichier précédemment générer.

Pour que l'appel se lance automatiquement le fichier doit avoir l'user Asterisk et non root car cela empêche au fichier d'être ouvert automatiquement par Asterisk et par conséquent son exécution. Pour cela on doit utiliser la commande :

```
chown asterisk:asterisk /var/spool/asterisk/outgoing/nomDuFichier.call
```

Dans le cadre de l'application ces commandes seront remplacées par des fonctions de la classe QFile, classe qui permet de manipuler les fichiers

- la copie :bool copy (const QString &newName);
- le changement de permission : bool setPermissions (Permissions permissions);

En cas d'erreur :

les erreurs liées aux appels passés automatiquement sont stockées dans le fichier **full** dans le répertoire :

/var/log/asterisk/

Après l'exécution de l'appel, le fichier est supprimé automatiquement du répertoire .

Voici un exemple du contenu du fichier "full", comme on peut le voir le système n'arrive pas à ouvrir le fichier appel2.call et n'arrive donc pas à l'exécuter :

```
[Jan 31 13:46:29] WARNING[2687] pbx_spool.c: Unable to open /var/spool/asterisk/outgoing/appel2.call: No such file or directory, deleting
[Jan 31 13:46:29] WARNING[2687] pbx_spool.c: Failed to scan service '/var/spool/asterisk/outgoing/appel2.call'
[Jan 31 13:46:30] WARNING[2687] pbx_spool.c: Unable to open /var/spool/asterisk/outgoing/appel2.call: No such file or directory, deleting
[Jan 31 13:46:30] WARNING[2687] pbx_spool.c: Failed to scan service '/var/spool/asterisk/outgoing/appel2.call'
[Jan 31 13:46:31] WARNING[2687] pbx_spool.c: Unable to open /var/spool/asterisk/outgoing/appel2.call: No such file or directory, deleting
[Jan 31 13:46:31] WARNING[2687] pbx_spool.c: Failed to scan service '/var/spool/asterisk/outgoing/appel2.call'
[Jan 31 13:46:32] WARNING[2687] pbx_spool.c: Unable to open /var/spool/asterisk/outgoing/appel2.call: No such file or directory, deleting
[Jan 31 13:46:32] WARNING[2687] pbx_spool.c: Failed to scan service '/var/spool/asterisk/outgoing/appel2.call'
[Jan 31 13:46:33] WARNING[2687] pbx_spool.c: Unable to open /var/spool/asterisk/outgoing/appel2.call: No such file or directory, deleting
[Jan 31 13:46:33] WARNING[2687] pbx_spool.c: Failed to scan service '/var/spool/asterisk/outgoing/appel2.call'
[trixbox2.localdomain asterisk]#
```

On peut voir que le fichier appel2.call n'a pas pu être ouvert et qu'il n’a pas été exécuté.

4.3 Échange de données

4.3.1 Étude des solutions possibles

Un échange de données s'effectue entre la personne qui va générer la campagne et celle qui va l'analyser afin de pouvoir stopper la campagne si la qualité de service (QoS) n'est pas convenable. Ces deux cas seront réalisés sur la même machine, mais avec deux applications différentes. Nous avions pensé à utiliser les tubes nommés.

Cette idée a été abandonnée parce qu'elle ne convenait pas aux attentes convenues, car le but étant de s'échanger un événement qui déclenchera l'arrêt de la campagne et les tubes n'offrent pas cette possibilité. Nous nous sommes donc tournés vers les signaux.

Un signal : est un événement généré par le système en réponse à certaines conditions et dont l'envoi à un processus peut déclencher une réaction.C'est pour ceci que la génération des appels se déroulera en parallèle de l'analyse de la campagne.

L'application qui analyse la qualité de service devra envoyer un signal à l'application qui génère les appels, ce signal sera traité par un slot priver qui devra stopper la campagne.

Un slot : c'est la fonction qui est appelée lorsqu'un événement s'est produit. On dit que le signal appelle le slot. Concrètement, un slot est une méthode d'une classe.

4.4 Choix concernant le développement

4.4.1 Choix liés aux développements de l'application

4.4.1.a Langage utilisé

Le C++ nous parait le langage le mieux adapte a nos besoins. Il est orienté objet. C'est le 3e langage le plus utilise au monde, sa communauté est donc évidemment très importante. Nous avions déjà une connaissance de ce langage.

Ce langage répond à toute les contraintes qui nous sont données et il propose des solutions adaptées pour réaliser cette application de manière convenable.

4.4.1.b Bibliothèque utilisée

QT est un framework oriente objet et développe en C++. Il offre des composants d'interface graphique, de gestion des fils d'exécution, d'accès de données et bien d'autres choses. Nous avions déjà utilisé des bibliothèques QT au cours de notre formation, ce qui facilitait l'accès et l'apprentissage. Le système de signaux et slots est donc déjà connu. QT permet une portabilité par simple recompilation du code source. Il est également utilisé par un grand nombre de personnes, sa communauté est donc active et importante. La documentation étant importante et complète elle nous permet de

faire face à la plupart des problèmes que nous pourrions rencontrer

Notre choix s'est donc dirigé très rapidement vers QT.

4.4.1.c Choix de l'EDI

Afin de développer l'application, QtCreator est utilisé, il est multiplateforme, ce qui facilite le transfert en cas de changement de système d'exploitation. Nous aurions pu également utiliser Qdevelop ou encore des EDI possibles comme Eclipse ou Visual Studio en y ajoutant un pluging QT. QtCreator est vraiment simple d'utilisation et est plus léger qu'un EDI comme Eclipse ou Visual. La prise en main est immédiate, ayant déjà utilisé QT .

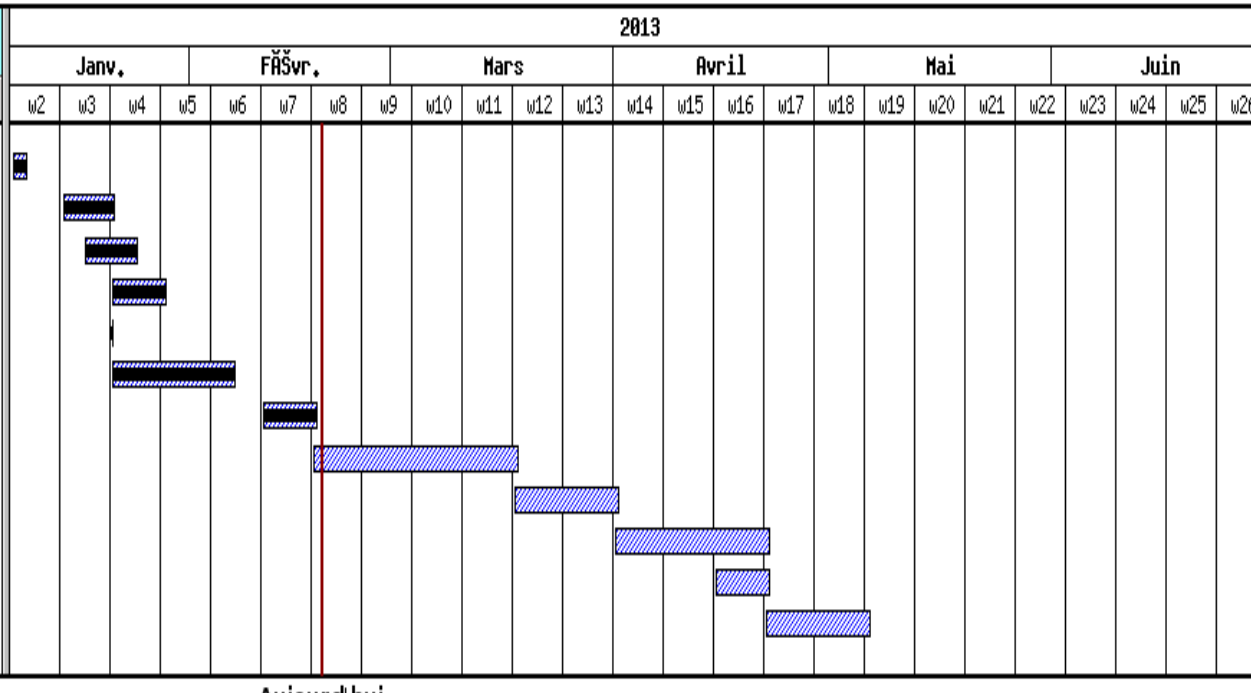
4.4.1.d Système d'exploitation

Le cahier des charges n'émet aucune contrainte vis a vis du système d'exploitation donc l'application devra aussi bien fonctionné sous Windows comme sous Linux.

Voici mes prévisions pour a suite du projet elles sont susceptible de changer en fonction de l'avancement du projet et des problèmes rencontrer.

Pondeuse d'appels

Nom de tâche	Dur.	Démarrer	fin
lecture du cahier des charges	16 h	07/Jan/2013	09/Jan/2013
description d'un appel	48 h	14/Jan/2013	21/Jan/2013
renseignement sur la téléphonie	48 h	17/Jan/2013	24/Jan/2013
diagramme de classe	48 h	21/Jan/2013	28/Jan/2013
mise en place de la pondeuse d'ap.	7 h	21/Jan/2013	21/Jan/2013
rédaction du dossier pour la revu.	112 h	21/Jan/2013	07/Feb/2013
précisé les diagrammes	48 h	11/Feb/2013	18/Feb/2013
codage des threads	168 h	18/Feb/2013	18/Mar/2013
choix et presentation d'un test	88 h	18/Mar/2013	01/Apr/2013
codage des diférente classe	128 h	01/Apr/2013	22/Apr/2013
intégration	48 h	15/Apr/2013	22/Apr/2013
correction des modules	88 h	22/Apr/2013	06/May/2013



5. Conception préliminaire : Partie Dauvergne Florian (E3)

5.1 Qualité de service :

5.2 Remise en contexte

Pour rappel, un nombre prédéfini d'appels va être généré que ce soit pour tester le centre d'appels ou encore tester le nombre d'appels reçus par les lignes opérateurs afin de savoir si leur capacité est bien celle prévue.

Néanmoins, il convient d'analyser la qualité de ces appels, car bien que ceux-ci soient acheminés d'un interlocuteur à l'autre, il se peut que la communication ne soit pas aisée, en cas de très mauvaise qualité impossible, cela peut être dû au temps d'acheminement ou à l'écho présent lors d'un appel.

Cette analyse détermine donc la qualité de service

5.3 Qu'est-ce que la qualité de service ?

La qualité de service (QDS) ou Quality of service (QoS) est la capacité à véhiculer dans de bonnes conditions un type de trafic donné, en matière de disponibilité, débit, délais de transmission, gigue, taux de perte de paquet sur un réseau. La qualité de service désigne la capacité d'un service à répondre par ses caractéristiques aux différents besoins de ses utilisateurs ou consommateurs.

La qualité de service est un concept de gestion qui a pour but d'optimiser les ressources d'un réseau et de garantir de bonnes performances aux applications. La qualité de service permet d'offrir aux utilisateurs des débits et des temps de réponse différenciés par applications (ou activités) suivant les protocoles mis en œuvre au niveau de la structure. Selon le type du service envisagé (ici la téléphonie), la qualité pourra résider dans la qualité de codage (compression de la voix), le délai (délai d'acheminement de l'appel), la gigue (fluctuation du signal), l'Écho ou encore le taux de pertes de paquets (pertes sans influences pour la voix ou perte quasi nulle).

5.4 Paramètres QoS

Les aspects déterminants pour la qualité de la voix sur un réseau sont le traitement de la voix, la clarté, le délai de bout en bout et l'écho. Ils dépendent des différents composants de la chaîne de transmission, de leur paramétrage, de l'architecture générale de la chaîne, et dans le cas de la VoIP des flux concurrents.

Ces aspects sont les suivants :

- Traitement de la voix : lors de l'émission du signal, la voix est traitée, c'est à dire codée et éventuellement compressée, avant d'être transmise.

- La Gigue, elle représente la fluctuation du signal numérique, dans le temps ou

en phase. La gigue est la variation de la latence et ne dépend pas donc pas de la latence. Vous pouvez avoir de hautes latences et une gigue très basse.

• Le délai d'acheminement de bout en bout est le temps de propagation de la voix à travers le réseau de l'émetteur vers le récepteur.

• L'écho est le son émis par l'émetteur qui lui revient.

• La perte de paquets, elle correspond à la non-délivrance d'un paquet de données.

Qualité de codage de la voix :

Les différents échantillonnages :

Le paramètre d'échantillonnage ou codec (pour compression/décompression) est structurant en Voie. Le codec détermine à quelle vitesse la voix est échantillonnée et dimensionne par la même le flux de données numériques que va générer la transformation d'un échantillon temporel de voix analogique. Les codecs sont répertoriés par leur nom à l'ITU.

L'échantillonnage consiste à relever à intervalle régulier la valeur du signal qui représente le son. Plus cette valeur est élevée plus la qualité est bonne et la consommation de bande passante importante.

Les codecs les plus utilisés et leurs vitesses d'échantillonnage sont les suivants :

G.711	64 kb/s	Codage sans perte, utilisation importante de bande passante.
G.726	32 kb/s	Codage avec pertes, division par deux de la bande passante par rapport à G 711.
G.729	8 kb/s	Faible consommation de la bande passante, codage avec pertes mais utilisation possible pour la téléphonie.
G.723.1 MPMLQ	6.3 kb/s	Très faible utilisation de bande passante, codage avec pertes. La musique et les codes DTMF ne peuvent pas être transportés de manière fiable par ce codec

G.723.1 ACELP	5.3 kb/s	Très faible utilisation de bande passante, codage avec pertes. La musique et les codes DTMF ne peuvent pas être transportés de manière fiable par ce codec

Le choix du codec est un compromis entre la qualité de service souhaitée et la capacité de l'infrastructure IP à délivrer une bande passante et des paramètres de QoS qui vont impacter cette qualité. Le paramètre le plus déterminant auquel on s'intéresse pour commencer est la bande passante que l'on met en regard du nombre de communications simultanées à écouler.

C'est une des raisons pour laquelle le choix du codec impacte la clarté de la voix, indépendamment des autres caractéristiques de l'infrastructure. Plus la fréquence d'échantillonnage est grande, meilleure sera la numérisation.

Sauf à être contraint par un besoin impératif de gain de bande passante, on choisira le codec G.711 qui assure la numérisation la plus rapide de la voix ainsi que la plus grande clarté pour le récepteur.

Délai d'acheminement :

Le délai de transit est un des paramètres influençant fortement la QoS d'un service de voix sur IP. C'est le temps que va mettre en moyenne un paquet IP contenant un échantillon de voix pour traverser l'infrastructure entre deux interlocuteurs. Ce temps de transit comporte quatre composantes :

- Le délai d'échantillonnage
- Le délai de propagation
- Le délai de transport
- Le délai des mémoires tampons de gigue

Le délai d'échantillonnage est la durée de numérisation de la voix à l'émission puis de conversion en signal voix à la réception. Ce temps dépend du type de codec choisi et varie de quelques millisecondes avec le codec G.711 (échantillonnage 64 kb/s) à plus de 50 ms en G.723 (échantillonnage 6,3 ou 5,3 kb/s). C'est une des raisons pour laquelle le choix du codec impacte le score MOS d'appréciation de la clarté de la voix, indépendamment des autres caractéristiques de l'infrastructure.

Le délai de propagation est la durée de transmission en ligne des données numérisées. Cette durée est normalement très faible par rapport aux autres composantes du délai de transit, de l'ordre de quelques millisecondes.

Le délai de transport est la durée passée à traverser les routeurs, les commutateurs et les autres composants du réseau et de l'infrastructure de téléphonie IP. L'ordre de grandeur est
de plusieurs dizaines de millisecondes, voir centaines de millisecondes.

Le délai des buffers de gigue est le retard introduit à la réception en vue de lisser la variation de temps de transit, et donc de réduire la gigue de phase. L'ordre de grandeur est de 50 ms.
Les éléments d'infrastructure, notamment les routeurs, peuvent également mettre en œuvre des mémoires tampons de gigue.

La gigue de phase :
La variation de temps de transit, ou gigue de phase est la conséquence du fait que tous les paquets contenant des échantillons de voix ne vont pas traverser le réseau à la même vitesse. Cela crée une déformation de la voix ou un hachage.

La gigue de phase est indépendante du délai de transit. Le délai peut être court et la gigue importante ou inversement. La gigue est une conséquence de congestions passagères sur le réseau, ce dernier ne pouvant plus transporter les données de manière constante dans le temps. La valeur de la gigue va de quelques ms à quelques dizaines de millisecondes.

La perte de données :
La transmission de la voix par paquets s'appuie sur le protocole RTP (real-time transport protocole). Ce dernier permet de transmettre sur IP les paquets de voix en reconstituant les informations même si la couche de transport change l'ordre des paquets. Il utilise pour cela des numéros de séquence et s'appuie sur UDP.
Les contraintes temps réel de délai de transit évoqué plus haut rendent inutile la retransmission des paquets perdus : même retransmis un datagramme RTP arriverait bien trop tard pour être d'une quelconque utilité dans le processus de reconstitution de la voix. En voix sur IP, on ne retransmet donc pas les données perdues. Ces pertes de données VoIP sont dues aux congestions sur le réseau, entraînant des rejets de paquets tout au long du réseau, ou à une gigue excessive qui va provoquer des rejets de paquet dans les buffers de gigue du récepteur, ceux-ci ne pouvant pas accueillir tous les paquets arrivés en retard.

Une perte de données régulière, mais faible est moins gênante en voix sur IP que des pics de perte de paquets espacés, mais élevés. En effet, l'écoute humaine s'habituera à une qualité moyenne, mais constante et en revanche supportera peu de soudaines dégradations de la QoS. Le taux de perte en VoIP est typiquement de quelques pour cent ou dixièmes de pour cent.

Recommandations générales pour assurer la qualité de la voix sur IP :

	Bon	Moyen	Mauvais
Délai d'acheminement	D < 150 ms	150 ms < D < 400 ms	400 ms < D
Gigue de phase	G < 4 %	4 % < G < 7 %	7 % < G
Perte de données	P < 1 %	1 % < P < 3 %	3 % < P
Écho	E < 1 %	1 % < E < 5 %	5 % < E

Une qualité d'appel moyenne n'entraînera pas un arrêt de la campagne d'appels puisque l'on considère que les deux interlocuteurs sont en mesure de communiquer. Cela sera donc à titre d'information sur la limite probable du nombre maximum d'appels possibles avant que la qualité devienne trop mauvaise.
Une qualité de voix mauvaise entraînera quant à elle, l'arrêt de la campagne d'appels puisque même si les appels sont acheminés étant donné que l'écho ou encore la perte de données sont trop importants la communication n'est pas possible.

5.5 Les logiciels d'analyse de la QoS

Afin d'analyser la communication de notre pondeuse d'appels, j'ai d'abord cherché s'il existait des fonctions dans Asterisk puisque nous utilisons trixbox. N'ayant rien trouvé je me suis tourné vers des logiciels d'analyse existant j'ai ainsi sélectionné et comparé ceux qui me semblait les plus intéressants.

Trois outils gratuits et open source fournissent une précieuse aide pour mesurer la qualité d'un lien réseau.

Wireshark,

C'est l'analyseur réseau le plus utilisé.
Il fournit un grand nombre d'informations sur la VoIP comme la gigue.Wireshark est l'analyseur réseau le plus populaire du monde. Cet outil extrêmement puissant fournit des informations sur des protocoles réseau et applicatives à partir de données capturées sur réseau.Comme un grand nombre deprogrammes, Wireshark utilise la librairie réseau Pcap pour capturer les paquets.

La force de Wireshark vient de :
- sa facilité d'installation.
- sa simplicité d'utilisation de son interface graphique.
- son très grand nombre de fonctionnalités.

Wireshark fut appelé Ethereal jusqu'en 2006 où le développeur principal décida de changer son nom en raison de problèmes de copyright avec le nom de Ethereal qui était enregistré par la société qu'il décida de quitter en 2006.

WANem : WANem

Un outil localisé entre deux hôtes comme un téléphone et un PBX pour simuler une qualité de lien réseau spécifique. Des paramètres comme la latence, la bande passante, la perte de paquet et la gigue sont disponibles.

Comme WANem est "au milieu", le routage doit être configuré sur les deux machines de test pour forcer le trafic entre eux à passer à travers WANem.
Un grand nombre de paramètres comme la taille de la bande passante, la perte de paquet, la gigue, la latence peuvent être configurés avec une interface web efficace.

WANem doit être situé entre deux hôtes entre lesquels nous voulons simuler un lien réseau. Les paramètres de routage qui ont besoin d'être sur les clients ou sur la machine WANem restent extrêmement faciles.

Il n'est pas nécessaire d'être un expert réseau et Linux pour utiliser WANem. En effet, le lancement, les configurations réseau et les tests sont particulièrement faciles et très rapides.

Iperf :

Iperf est premièrement utilisé pour mesurer la bande passante disponible entre deux hôtes sur lesquels tourne Iperf. Il est également possible de mesurer la gigue et la perte de paquet.

Iperf est un outil pour mesurer la bande passante et la qualité d'un lien réseau. Ce dernier est délimité par deux machines sur lesquelles est installé Iperf.

Iperf utilise les différentes propriétés de TCP et d'UDP pour fournir des statistiques sur des liens réseau. Iperf peut être installé très facilement sur n'importe quel système UNIX/Linux ou Microsoft Windows. Un hôte doit être configuré en tant que client et l'autre en tant que serveur.

Tableau comparatif :

Besoins	Plusieurs critères entraient en compte dans le choix du logiciel, il devait être gratuit, relativement simple d'utilisation ou bien documenté et prendre en compte nos paramètres à analysés qui sont le délai d'acheminement, la perte de données, la gigue de phase et l'écho.		
Logiciels	WANem	Iperf	Wireshark
Paramètres analysés	Bande passante, perte de paquets et gigue.	Latence gigue et perte de paquets	Écho, latence, gigue et pertes de paquets.
Fonctionnement	WANem doit être situé entre deux hôtes entre lesquels nous voulons simuler un lien réseau.	Iperf doit être lancé sur deux machines se trouvant de part et d'autre du réseau à tester. L'une en mode serveur l'autre en mode client	Lancé en parallèle du réseau que l'on veut analyser.

Avantages	Simple d'utilisation de par son interface	Installation simple sur n'importe quel système Linux ou Windows.	Beaucoup de documentation Tous les paramètres sont pris en compte. Simple d'utilisation
Inconvénients	Simulation exclusivement Peu de paramètres pris en compte	Peu de documentation sur son fonctionnement et son utilisation. Il ne mesure pas l'écho.	
Prix	Gratuit	Gratuit	Gratuit

Pour résumer :

Tous ces logiciels permettent d'analyser les différents paramètres déterminant la qualité de service d'une campagne d'appels. Tous ne permettent pas d'analyser tous les paramètres et ne sont pas simples d'utilisation. Wireshark est le plus performant de nos 3 logiciels choisis puisqu'il est le plus complet en terme de paramètres analysés.

5.6 La librairie Pcap

Néanmoins Wireshark serait trop lourd à interfacer avec notre application. Il n'est donc pas utilisable telle qu'elle, c'est pourquoi on s'est intéressés à son fonctionnement. Wireshark utilise sa propre interface graphique et fonctionne sur la librairie Pcap afin d'analyser la communication. Je vais donc utiliser cette librairie, que je vais intégrer à mon interface QT afin de coder mon application.

Je vais néanmoins utiliser Wireshark comme point de comparaison dans le but de vérifier le bon fonctionnement de mon application et de ses résultats obtenus.

Présentation générale de PCAP :

PCAP est une interface de programmation permettant de capturer un trafic réseau. Elle est implémentée sous les systèmes GNU/Linux, FreeBsd, NETBSD, OpenBSD et MAC OS X par la bibliothèque Libpcap. WinPcap est le portage sous Windows de Libpcap.
Les outils de supervision réseau peuvent utiliser Pcap et/ou WinPcap afin de capturer les paquets transitant sur le réseau. Les versions récentes permettent également de transmettre des paquets sur la couche liaison.

Libpcap et WinPcap permettent aussi de sauvegarder les paquets capturés dans un fichier, et la lecture de fichiers provenant de captures précédentes. Des applications peuvent être développées utilisant Libpcap ou WinPcap pour pouvoir capturer du trafic réseau, le lire, l'enregistrer et l'analyser.

Libpcap et WinPcap fournissent l'outil de filtrage et de capture de paquets qu'utilisent de nombreux logiciels libres ou commerciaux d'analyse de trafic, allant des analyseurs de protocoles, moniteurs réseau aux systèmes de détection d'intrusion, générateurs de trafic et testeurs de réseau.

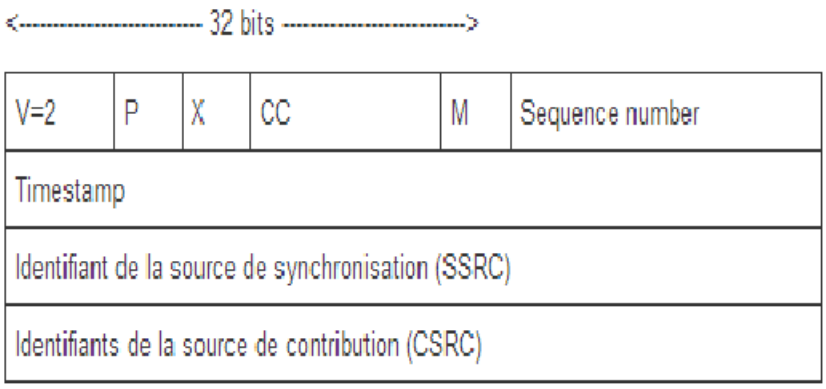
PCAP permet de faire des captures de trafic en appliquant des filtres à la capture. On retrouve des filtres par adresse IP (source, destination, les 2), par adresses Ethernet, par protocole (de n'importe quelle couche), etc.

Quand on parle de capturer le trafic, cela veut dire enregistrer tout ce que remonte la carte réseau. Étudier le comportement d'un réseau en analysant 750 000 trames devient très compliqués et fastidieux.oin.

Pour ce faire suivant les fonctions PCAP nous filtrerons suivant certains protocoles par exemple SIP, RTP et RTCP. Nous capturerons les informations présentes dans les en têtes de ces paquets.

Nous devons dans l'analyse de la communication calculer la gigue afin de nous assurer que celle-ci n'est pas trop élevée. Cette opération peut être réalisés en analysant l'en tête d'un paquet RTP.

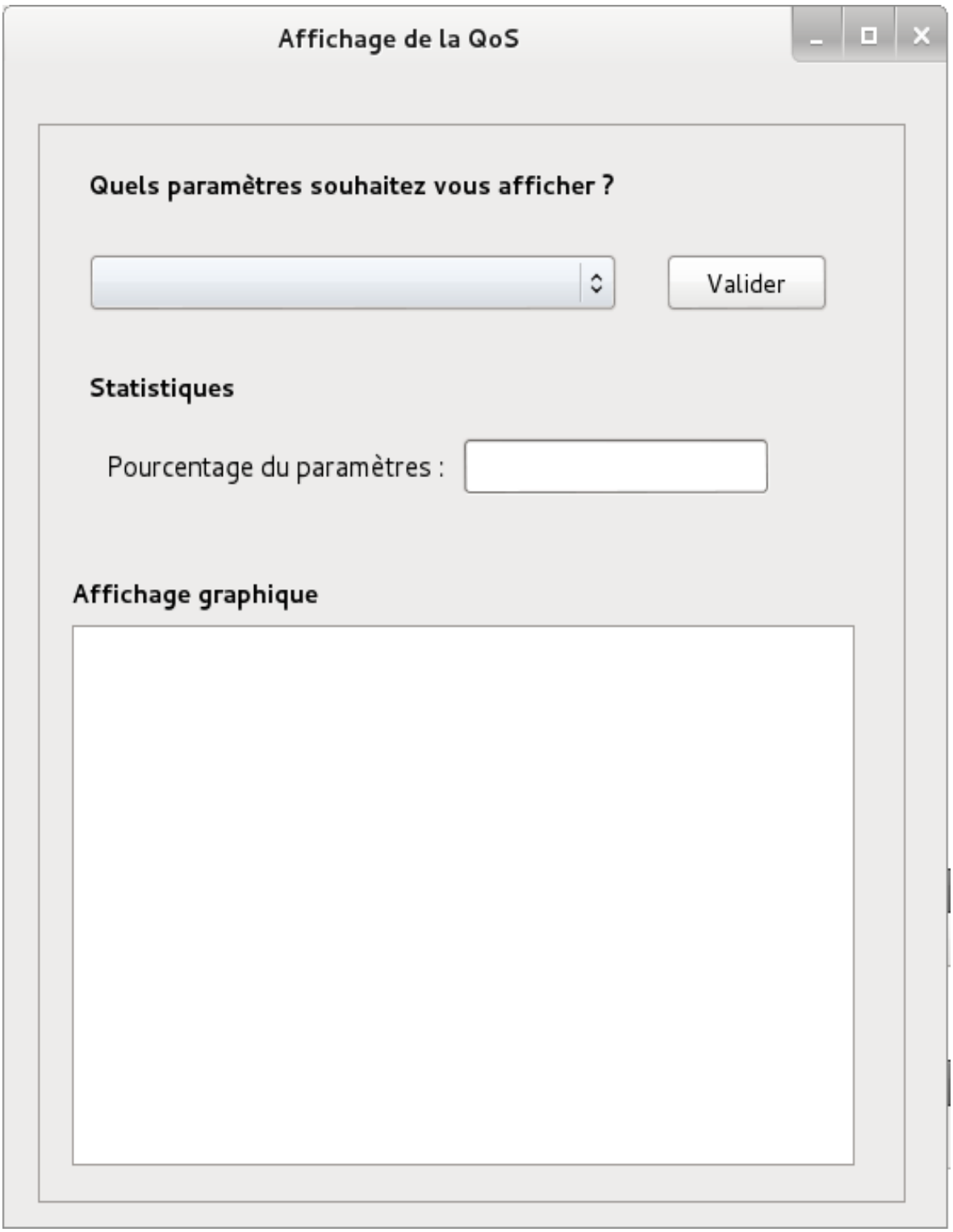
Ici l'en tête d'un paquet RTP :



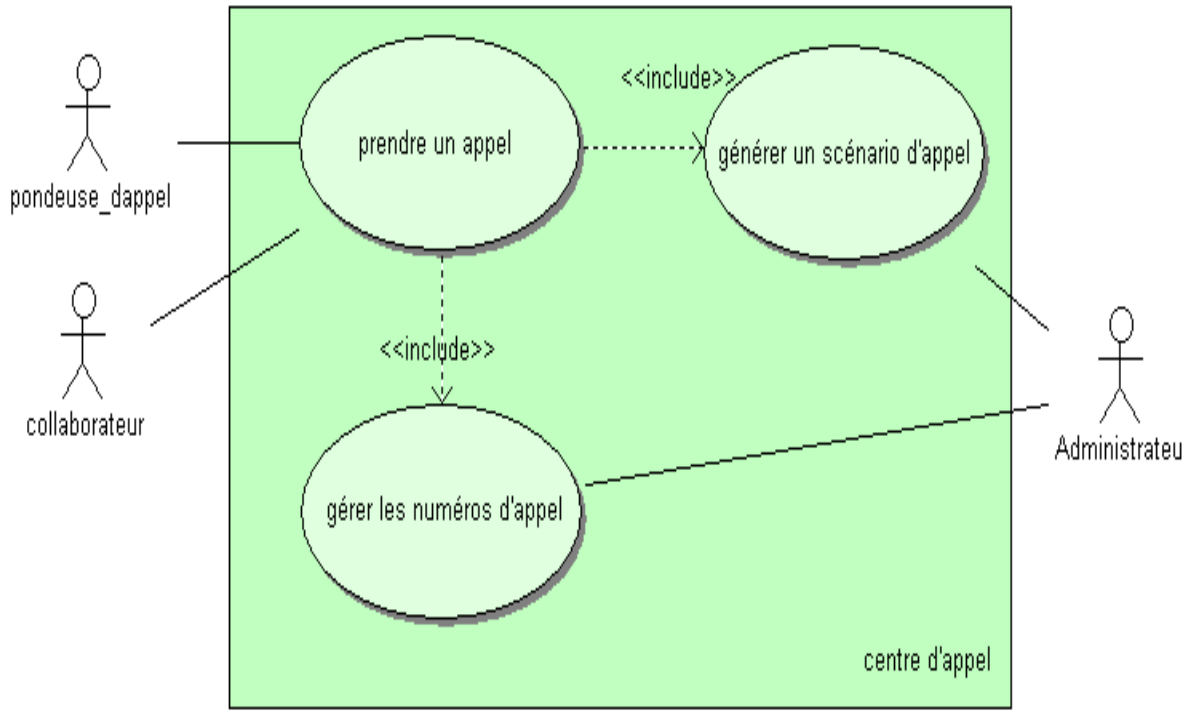
Le champ timestamp : 32 bits reflètent l'instant où le premier octet du paquet RTP a été échantillonné. Il permet le calcul de la gigue à la destination.

5.7 Affichage de la QoS

Une fois les différents paramètres de la qualité de service analysés et enregistrés, l'utilisateur pourra afficher celui qu'il souhaite. C'est-à-dire qu'il aura le choix du paramètre qu'il souhaite voir. Une fois qu'il aura choisi celui-ci, il aura les données numériques de ce paramètre (par exemple le temps en millisecondes du délai d'acheminement. Puis un graphique de la variation de ce délai au cours du temps.


Affichage de la QoS
Quels paramètres souhaitez vous afficher ?
Valider
Statistiques
Pourcentage du paramètres :
Affichage graphique

6. Conception détaillée : Partie Prieur Quentin (E4)

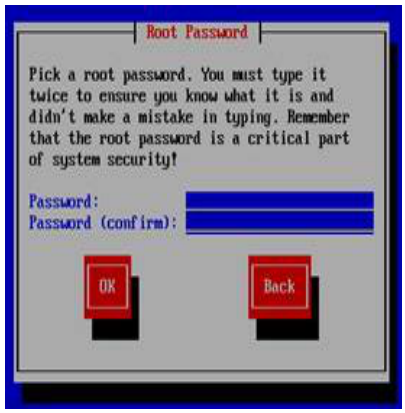
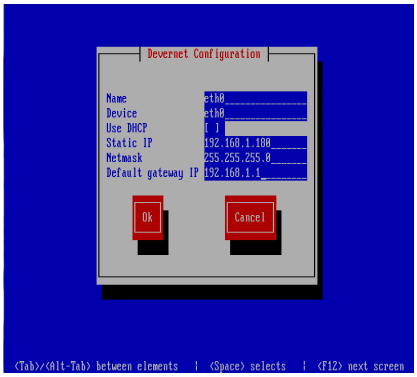


Je suis l'étudiant 4, en vert sur le diagramme. Je suis en charge de la partie centre d'appels.

6.1 Installation du serveur trixbox

Il a fallu installer l'os qui va servir à héberger notre pondeuse d'appels. Celui-ci était donné, il s'agit de Trixbox. C'est un OS basé sur une distribution Linux Cent OS, avec une surcouche nommée Asterisk, qui est dédiée à la téléphonie.

Voici en détail son installation :

<table>
<tr><td>
</td><td>
</td><td> 
</td></tr>
<tr><td>Choix de la langue du clavier</td><td>Choix de la localisation du serveur</td><td>Choix du mot de passe root</td></tr>
<tr><td>
</td><td> 
</td><td></td></tr>
<tr><td colspan="3">Configuration des interfaces réseau</td></tr>
</table>

6.2 Serveur FTP

Pour pouvoir exécuter les scripts qui nous permettront de lancer les différents appels, nous devons installer un serveur FTP sur le quelle nous enverrons les fichiers .call à distance. Dans un second temps il servira aussi à modifier le fichier *extension.conf* qui gère les différents scénarios d'appels.

6.2.1 Installation et configuration

Pour installer le service FTP sur notre serveur (de base CENTOS), nous avons besoin du paquet *vsftpd*. L'installation passe par la commande :

su yum install vsftpd

En ce qui concerne la configuration, elle se fait dans le fichier */etc/vsftpd /vsftpd.conf*

Ce que nous avons dû modifier :

/* Connexion anonyme */

anonymous_enable=NO

Pour des raisons de sécurité, nous avons préféré interdire les connexions anonymes via le FTP

/* Autorisation des utilisateurs locaux*/

local_enable=YES

Dans un premier temps nous pensions accéder au FTP avec un compte local, comme le compte "admin" créé dans la partie "Ajout d'utilisateur système", il était donc nécessaire d'autoriser ces connexions.

/* Autorisation des écritures sur le serveur via FTP */

write_enable=YES

Le FTP nous servira principalement à créer et modifier des fichiers, nous avons donc besoin des droits d'écritures.

/* Permet de prcésisé le propriétaire l'upload */

chown_uploads=NO

Lorsque cette option est activée, tous les fichiers télédéchargés vers le serveur par des utilisateurs anonymes deviennent la propriété de l'utilisateur spécifié dans la directive `chown_username`. Ayant désactivé les accès anonymes cette option ne nous est pas utile.

/* Message d'accueil du FTP */

ftpd_banner=serveur FTP projet MMA

Il s'agit du message d'accueil du serveur FTP.

/* Bloque les users local a leur propre dossier */

chroot_local_user=NO

Cette option permet de bloquer les différents utilisateurs du FTP à leur dossier personnel. Les fichiers qui seront envoyer ou modifier sur le serveur seront dans le dossier /etc/asterisk, il ne s'agit donc pas du dossier personnel d'un utilisateur.

/* Liste des utilisateurs autorisés à se connecter via le FTP */

userlist_file=/etc/vsftpd.user_list

Ce champ indique le fichier liste qui contient les différents noms de compte qui on l'autorisation d'utiliser le serveur FTP.

6.2.1.a Ajout des utilisateurs au système

Pour pouvoir accéder au FTP, comme vue précédemment, les connexions anonymes étant refusées, nous aurons besoin d'un nom d'utilisateur et d'un mot de passe.

Voici la procédure d'ajout d'un utilisateur "admin" au système[6] (à exécuter en super utilisateur, en mode console) :

1. cmd -> *adduser* admin
2. cmd -> *passwd* admin
   - *Changing password for user admin*
   - *New UNIX password* : *****
   - *Retype new UNIX password* : ****
   - *passwd* : *All authentication* tokens updated successfylly

6.2.1.b Ajout des accès de connexion au FTP

De base, tous les utilisateur sauf ftpuser, n'ont pas d'accès au service FTP. Il faut les ajouter au fichier liste qui se trouve être, *vsftpd.user_list*.

1. Accéder a la liste des utilisateurs
   - cmd -> *vi (or vim) /etc/vsftpd.user_list*
2. Ajouter les utilisateurs qui doivent avoir un accès, ici "admin"

Exemple :

```
# vsftpd userlist
# If userlist_deny=NO, only allow users in this file
# If userlist_deny=YES (default), never allow users in this file, and
# do not even prompt for a password.
# Note that the default vsftpd pam config also checks /etc/vsftpd.ftpusers
# for users that are denied.
ftpuser
admin
```

3. Enregistrer et fermer le fichier.

6 Procédure de suppression d'utilisateur en annexe

6.2.1.c Autoriser les accès FTP à un compte virtuel : asterisk

Pour des raisons de droit d'accès aux fichiers, seul l'utilisateur "virtuel"[7] asterisk peut modifier les fichiers dont nous avons besoin pour générer des appels.

Nous avons donc choisi d'autoriser l'accès au serveur FTP via le compte *asterisk*. Les fichiers envoyés sont donc au même niveau d'autorisations, puisqu'ils seront exécutés par le même compte.

Pour ce faire nous avons du nous rendre dans le fichier *passwd* situer dans */etc/*

Avec la commande: *su vipw*

Le fichier /etc/passwd est composé de 7 champs qui sont séparés par ":".

Voici la signification de chacun des champs.

nom_du_compte : mot_de_passe : numero_utilisateur : numero_de_groupe : commentaire : répertoire : programme_de_demarrage

Sept champs sont explicités séparés par le caractère ":" :

- le **nom du compte** de l'utilisateur
- le **mot de passe** de l'utilisateur (codé bien sûr)
- l'**entier** qui identifie **l'utilisateur** pour le système d'exploitation (UID=User ID, identifiant utilisateur)
- l'**entier** qui identifie le **groupe** de l'utilisateur (GID=Group ID, identifiant de groupe)
- le **commentaire** dans lequel on peut retrouver des informations sur l'utilisateur ou simplement son nom réel
- le **répertoire de connexion** qui est celui dans lequel il se trouve après s'être connecté au système
- la **commande** est celle exécutée **après connexion** au système (c'est fréquemment un interpréteur de commandes)

Dans notre cas :

// Utilisateur sans possibilité de se connecter

asterisk:x:100:101::/var/lib/asterisk:/sbin/nologin

Pour autoriser un utilisateur de ce type à se connecter comme un utilisateur standard, il faut effectuer deux modifications.

Original : *asterisk:x:100:101::/var/lib/asterisk:/**s**bin/nologin*

Modifiée :*asterisk:x:100:101::/var/lib/asterisk:/bin/**bash***

7 Un compte virtuel est un compte qui n'existe pas réellement à la différence d'un compte local, il appartient en généralement à un service.

6.3 phpMyAdmin

Dans le cadre de la création et du suivi des campagnes d'appels, nous devons sauvegarder les différentes données. Pour cela nous avons choisi d'installer phpMyAdmin, qui nous permettra, de la création, jusqu'à la gestion et de stocker ces données dans une base MYSQL.

6.3.1.a Installation

La version de phpMyAdmin présente dans les dépôts, ne sont pas à jour, nous avons du procéder à une installation via un paquet .rpm

La procédure est la suivante, dans un terminal en mode super utilisateur :

```
wget http://pkgs.repoforge.org/phpmyadmin/phpmyadmin-2.11.9.6-1.el5.rf.noarch.rpm
rpm -i phpmyadmin-2.11.9.6-1.el5.rf.noarch.rpm
```

Une fois ceci fait, phpMyAdmin est installé, mais n'est pas forcement accessible, il suffit pour cela définir les autorisations sur son dossier.

```
chmod -R 755 /usr/share/phpmyadmin
```

chmod 755, donne au propriétaire tous les droits, aux membres du groupe et aux autres les droits de lecture et d'accès. C'est un droit utilisé traditionnellement sur les répertoires.

Voici un tableau qui reprend les différents droits :

Position binaire	Valeur octale	Les droits	Commentaire
000	0	- - -	Aucun droits
001	1	- - x	Executable
010	2	- w -	Ecriture
011	3	- w x	Ecrire et executer
100	4	r - -	Lire
101	5	r - x	Lire et executer
110	6	r w -	Lire et ecrire
111	7	r w x	Lire ecrire et executer

6.3.1.b Connexion à la base MYSQL

Pour que phpMyAdmin puisse accéder serveur MYSQL, il faut lui en donnée les droits.

Procédure :

Ouvrir en édition le fichier *config.inc.php* situer dans */etc/phpmyadmin*

vim /etc/phpmyadmin/config,inc,php

Original : *$cfg['Servers'][$i]['auth_type'] = 'cookie';*

Modifiée : *$cfg['Servers'][$i]['auth_type'] = 'http';*

La différence entre les deux est principalement dans la méthode d'authenfication. Dans la méthode originale, elle se fait à l'aide d'un stockage dans un cookie. Dans la seconde méthode elle se fait via le protocole http et n'est donc pas sauvegarder.

6.4 Trunk SIP : Mise en place

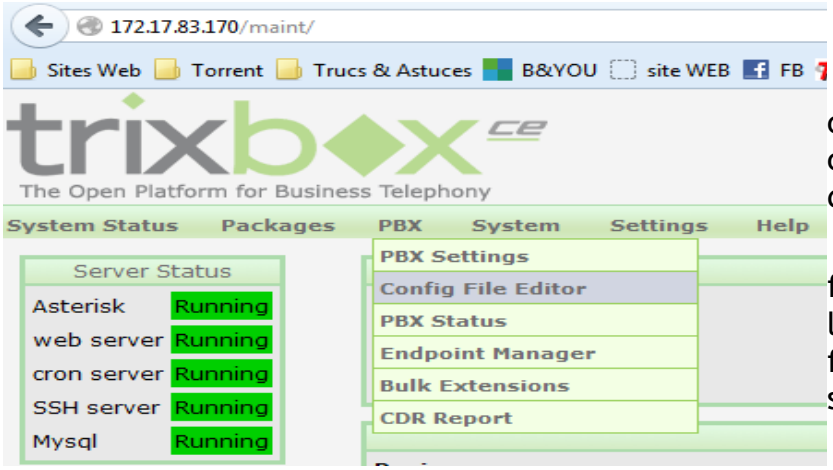
Dans notre cas, nous allons en avoir besoin pour faire le lien entre notre serveur Trixbox et L'OXE Alcatel pour pouvoir faire transiter des appels SIP de l'un vers l'autre, nous aurons en aurons donc besoin pour cette utilisation .

6.4.1 Solution N°1

Après avoir reçu le matériel prêté par les MMA, la solution était de mettre en place un Trunk SIP entre un serveur Trixbox et l'OXE Alacatel.

6.4.1.a Configuration : côté serveur Trixbox

Etape 1: Création du point de connexion



Le trunck SIP doit être créé comme un utilisateur, et cette création doit être faite à partir de l'interface de gestion web.

Dans un premier temps, il faut ajouter un utilisateur sur le serveur (ce qui modifiera le fichier de configuration sip.conf).

[OXEtoTrixbox] *// Nom du trunk*

type=friend,user,peer *// Type*

Peer:Une entité SIP pour Asterisk qui envoie des appels.
User: Une entité SIP qui emmet des appels à travers Asterisk.
Friend: Une entité qui est à la fois un user et un peer.

secret=toto *// Mot de passe*

context=mma

Un context, peut être comparer à une zone de travaille, seuls les utilisateurs du même context peuvent communiquer entre eux.

host=dynamic,nomDeDomaine,AdresseIP

dynamic: Cela signifie que n'importe quel hote peut se connecter.

insecure=port,invite / port / invite *// Type de sécurité*

port, invite: Ne nécessite pas de demande au préalable pour l'authentification, le port de connections est ignoré.

Invite: ne nécessite pas d'authentification des appels entrants, mais le port n'est pas ignorer

port : ignorer le numéro de port où la demande provenait

Configuration à ajouter dans le fichier sip.conf (etc/asterisk/sip.conf)

```
[OXEtoTrixbox]
type=friend
secret=toto
context=mma
host=dynamic
insecure=port,invite
```

Pour vérifier la bonne configuration, il faut afficher les points de connexions sur le serveur avec les commandes suivantes :

```
Serveur# rasterisk
*CLI> sip show peers
Name/username          Host            Dyn  Nat  ACL  Port   Status
OXEtoTrixbox                           D                5060   Unmoni
```

L'OXE Alcatel pourra à partir de maintenant se connecter avec les identifiants :

OXEtoTrixbox et le password : **toto**

Étape 2: Redirection des appels

Le serveur doit pouvoir rediriger les appels vers l'OXE Alcatel qui dispose d'un plan de numérotation interne tel que tous les numéros de poste, soit compris entre 2000 et 2999.

Pour cela, il faut modifier le fichier extentions.conf afin de définir correctement le contexte lié aux extensions et au trunck.

Configuration à ajouter dans le fichier extensions.conf

```
[mma]
exten => _2XXX,1,Dial(SIP/TrixboxtoOXE/${EXTEN})
```

_2XXX = (X peut varier de 0 à 9) donc numéro de 2000 à 2999.

1 = Il s'agit de la priorité

Dial = Il s'agit de l'action à effectuer, donc composer dans cotre cas.

SIP / = Protocole à utiliser

TrixboxtoOXE = Point de connexion

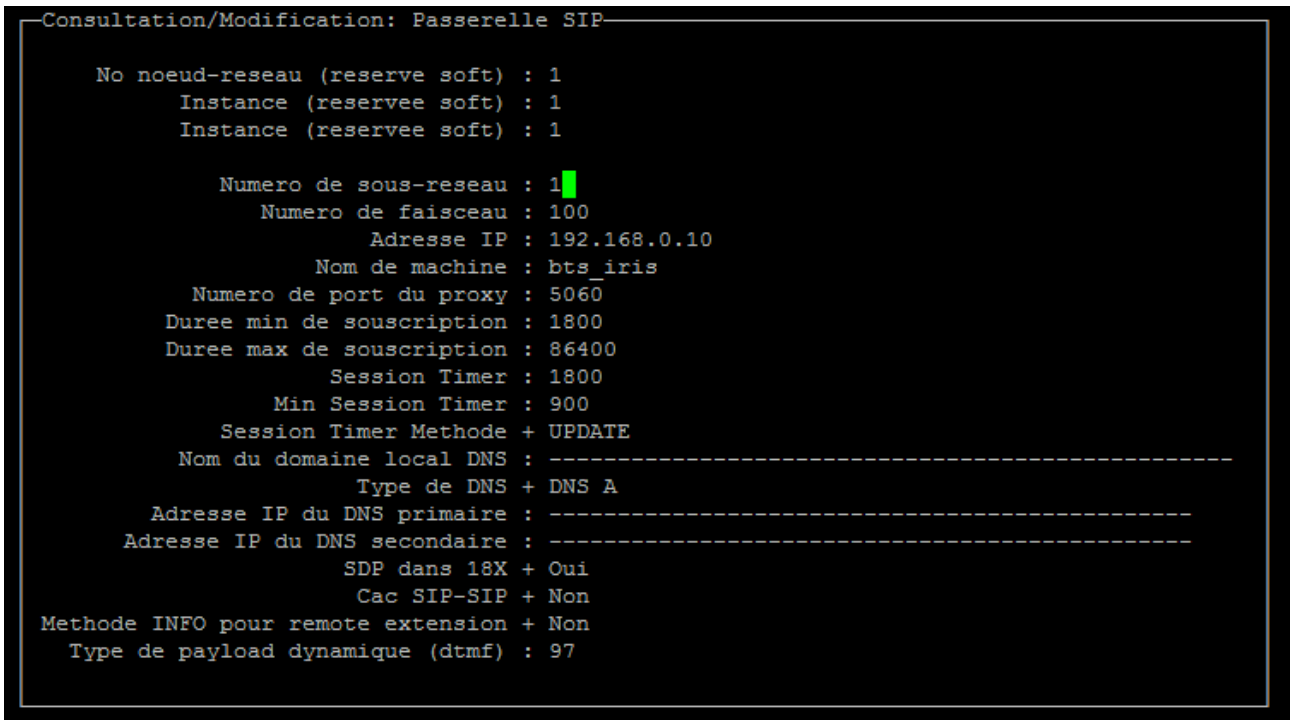
${EXTEN} = Expression remplacée par le numéro composé

6.4.1.b Configuration : côté OXE Alcatel

Il faut dans un premier temps se connecter via SSH sur l'OXE (192.168.0.10) avec les identifiants user : mtcl / passwd: mtcl. Puis exécuter la commande *mgr.*

Création d'une passerelle SPI :

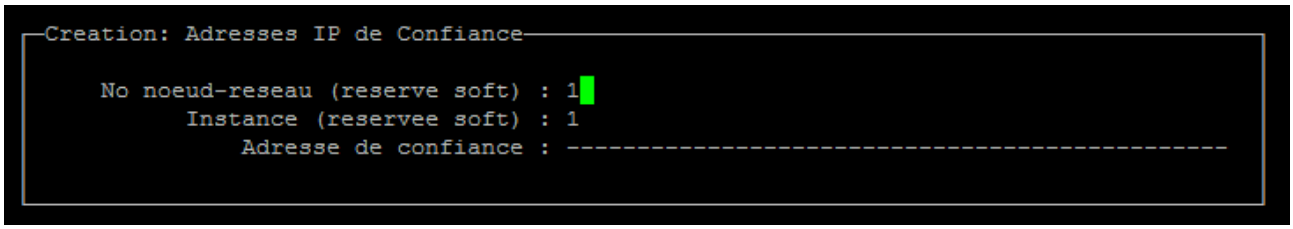
Se déplacer dans SIP > Passerelles SIP > Consultation/ Modification


Consultation/Modification: Passerelle SIP
No noeud-reseau (reserve soft) : 1
Instance (reservee soft) : 1
Instance (reservee soft) : 1
Numero de sous-reseau : 1
Numero de faisceau : 100
Adresse IP : 192.168.0.10
Nom de machine : bts_iris
Numero de port du proxy : 5060
Duree min de souscription : 1800
Duree max de souscription : 86400
Session Timer : 1800
Min Session Timer : 900
Session Timer Methode + UPDATE
Nom du domaine local DNS : ----------
Type de DNS + DNS A
Adresse IP du DNS primaire : ----------
Adresse IP du DNS secondaire : ----------
SDP dans 18X + Oui
Cac SIP-SIP + Non
Methode INFO pour remote extension + Non
Type de payload dynamique (dtmf) : 97

Il faut définir l'adresse de la passerelle SPI, dans notre cas, ça sera l'adresse le la CS 192.168.0.10 et valider par ctrl + V

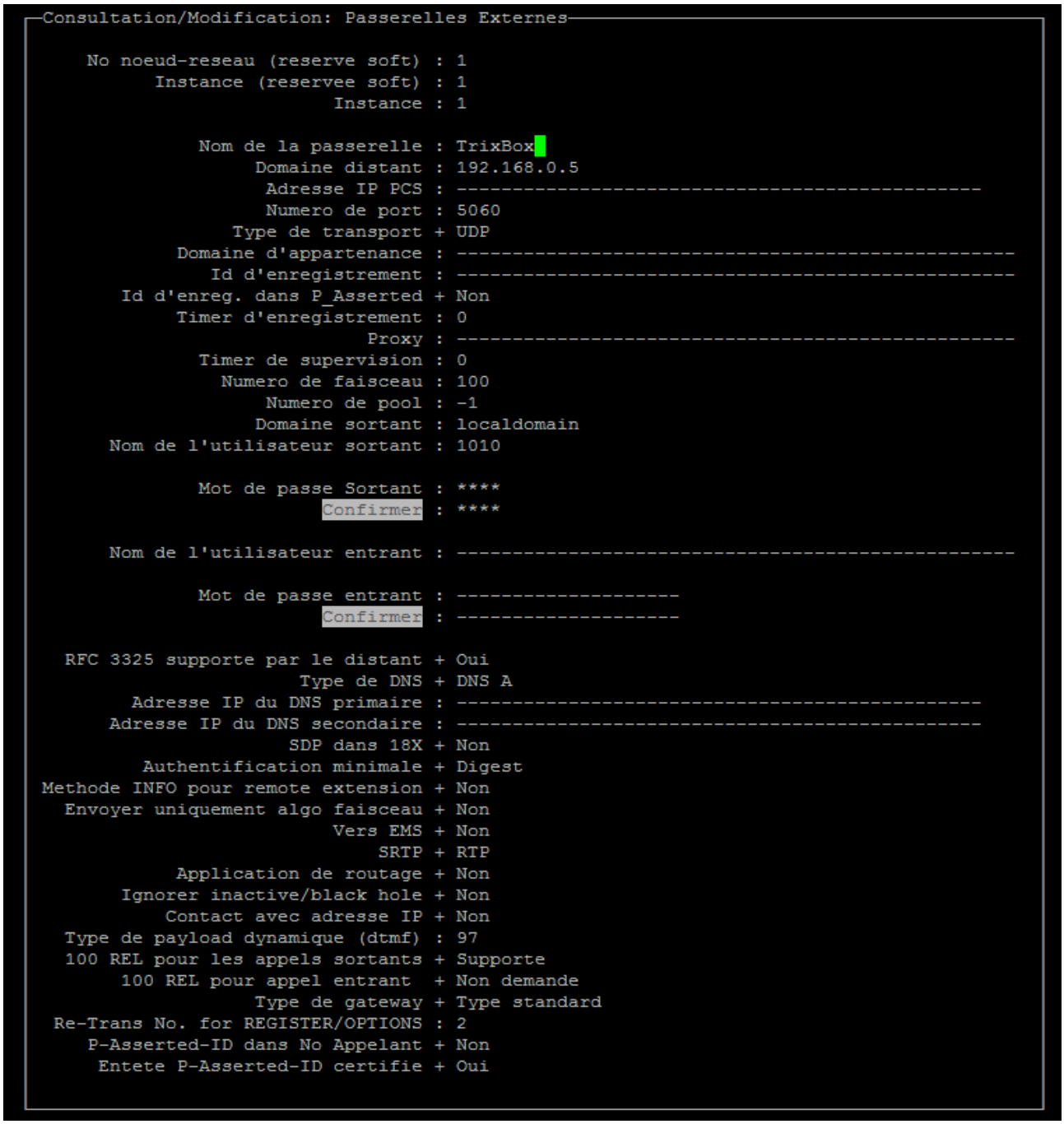
Ajout d'une adresse IP de confiance :

Se déplacer dans SIP > Adresse IP de confiance > Création



Il faut définir l'adresse de confiance dans notre cas, l'adresse IP locale de notre serveur Trixbox 192.168.0.5 et valider par ctrl + V

Configuration de la passerelle externe

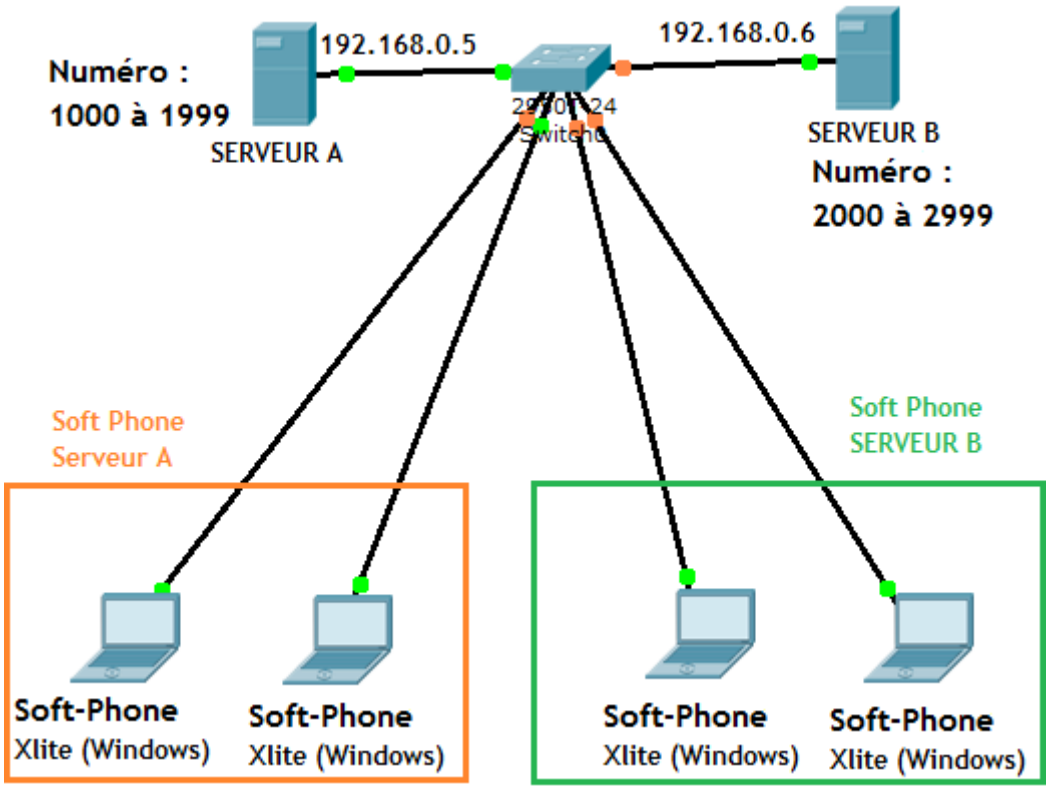

Consultation/Modification: Passerelles Externes
No noeud-reseau (reserve soft) : 1
Instance (reservee soft) : 1
Instance : 1
Nom de la passerelle : TrixBox
Domaine distant : 192.168.0.5
Adresse IP PCS : ----------
Numero de port : 5060
Type de transport + UDP
Domaine d'appartenance : ----------
Id d'enregistrement : ----------
Id d'enreg. dans P_Asserted + Non
Timer d'enregistrement : 0
Proxy : ----------
Timer de supervision : 0
Numero de faisceau : 100
Numero de pool : -1
Domaine sortant : localdomain
Nom de l'utilisateur sortant : 1010
Mot de passe Sortant : ****
Confirmer : ****
Nom de l'utilisateur entrant : ----------
Mot de passe entrant : ----------
Confirmer : ----------
RFC 3325 supporte par le distant + Oui
Type de DNS + DNS A
Adresse IP du DNS primaire : ----------
Adresse IP du DNS secondaire : ----------
SDP dans 18X + Non
Authentification minimale + Digest
Methode INFO pour remote extension + Non
Envoyer uniquement algo faisceau + Non
Vers EMS + Non
SRTP + RTP
Application de routage + Non
Ignorer inactive/black hole + Non
Contact avec adresse IP + Non
Type de payload dynamique (dtmf) : 97
100 REL pour les appels sortants + Supporte
100 REL pour appel entrant + Non demande
Type de gateway + Type standard
Re-Trans No. for REGISTER/OPTIONS : 2
P-Asserted-ID dans No Appelant + Non
Entete P-Asserted-ID certifie + Oui

Il faut définir plusieurs paramètres, le nom de la passerelle, le domaine à contacter, le numéro du port, le nom d'ulisateur sortant et son mot de passe.

6.4.1.c Conclusion :

Dans cette solution, les appels de l'oxe Alcatel faisait retentire la sonnerie mais ne permettait pas d'echanger une conversation, dans le sens inverse, un message vocal nous indiquait le numéro était invalide. De plus nous étions dans l'impossibilites de faire passer une communication entre les poste SPI et numeric de l'oxe et le serveur Trixbox.

6.4.2 Solution N°2



Vue que la solution précédente ne présentait des dificulté de fonctionnement, nous avons dû rechercher une solution intermèdiaire pour pouvoir continuer à avancer dans la suite du projet.

6.4.2.a Configuration: Mise en place du trunk

Le trunck SIP doit être créé comme un utilisateur, et cette création doit être faite à partir de l'interface de gestion web. Dans un premier temps, il faut ajouter un utilisateur sur le serveur (ce qui modifiera le fichier de configuration sip.conf).

Voici les config pour les 2 serveurs:

SERVEUR A	SERVEUR B
[trunck_B_vers_A]	***[trunck_A_vers_B]***
type=friend	***type=friend***
secret=toto	***secret=toto***
context=mma	***context=mma***
host=dynamic	***host=dynamic***
insecure=port,invite	***insecure=port,invite***

Une fois le point de connexion créé sur le serveur A, il faut que le serveur B s'enregistre. Dans le fichier sip.conf du serveur B il faut ajouter la ligne suivante (inversement pour le serveur A).

Serveur A	*register => **trunck_A_vers_B**:toto@192.168.0.6*
Serveur B	*register => **trunck_B_vers_A**:toto@192.168.0.5*

Pour vérifier la bonne configuration, il faut afficher l'état de la communication avec les commandes suivantes :

```
ServeurB#rasterisk
*CLI> sip show registry
Host                    Username          Refresh    State
192.168.0.5:5060        trunck_B_vers_A       105    Registered
```

6.4.2.b Configuration: redirection d'appels

Procédure de redirection des appels

Quand le serveur B s'est bien enregistré, le serveur A pourra rediriger les appels compris entre 2000 et 2999, vers le serveur B.

Pour cela, il faut modifier le fichier extentions.conf afin de définir correctement le contexte lié aux extensions et au trunck.

```
[mma]
exten => _2XXX,1,Dial(SIP/trunck_A_vers_B/${EXTEN})
```

Pour que le serveur B puisse rediriger les appels de 1000 à 1999 vers le serveur A il faut il ajouter les deux lignes suivant dans son fichier extensions.conf

```
[mma]
exten => _1XXX,1,Dial(SIP/trunck_B_vers_A/${EXTEN})
```

6.4.2.c Conclusion sur la solution N°1

La solution mise en place fonctionne, mais dans un seul sens B vers A. Elle n'est que très peut fiable les appels passe sans soucis particulier de temps à autre.

6.5 Conclusion

La téléphonie sur IP est devenue incontournable pour les équipementiers, les opérateurs, les entreprises et le grand public.

Si les enjeux économiques justifient largement cette convoitise, il ne faut cependant pas négliger les contraintes techniques à surmonter. (Par exemple, les soucis de compatibilité entre les systèmes Alcatel et Linux)

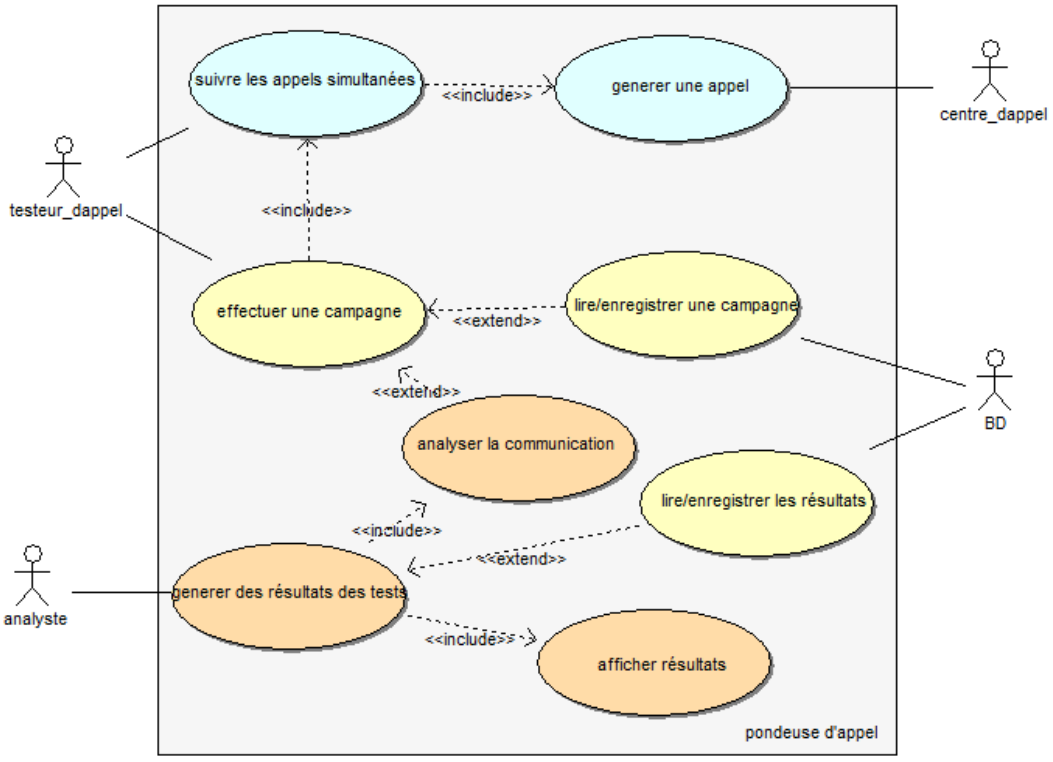
Durant le présent projet, il nous avait été confié la mission, pour l'entreprise MMA (Mutuelles du Mans Assurances) d'étudier et de mettre en place une pondeuse d'appels. Le serveur qui permettras d'emmètre les appels, les scénarios, et de sauvegarder toutes les données ont été réaliser et opérationnel. Le lien (Trunk SIP) est encore à étudier et à mettre en place.

Le travail de groupe ma permis de développer un sens du partage, nous avions une bonne ambiance pour conduire un bon projet.

Bien que quelques suprprise, comme les grand soucis de compatibilité et le grand nombre de changement de direction dans le projet, il a été fortement enrichissant pour moi dans plusieurs domaines :

- La découverte de la relation avec une personne tierce, extérieure au lycée, de l'entreprise MMA m'a permis de prendre connaissance des échanges possible avec le monde professionnel
- La découverte approfondie du réseau et de la téléphonie a été pour moi, une grande aide pour le choix de mon avenir professionnel. Pour l'avenir le métier d'administrateur réseau serait une voie intéressante (après une licence professionnelle pour approfondir mes connaissances).

7. Conception détaillée : Partie Plateau Mathias (E1)



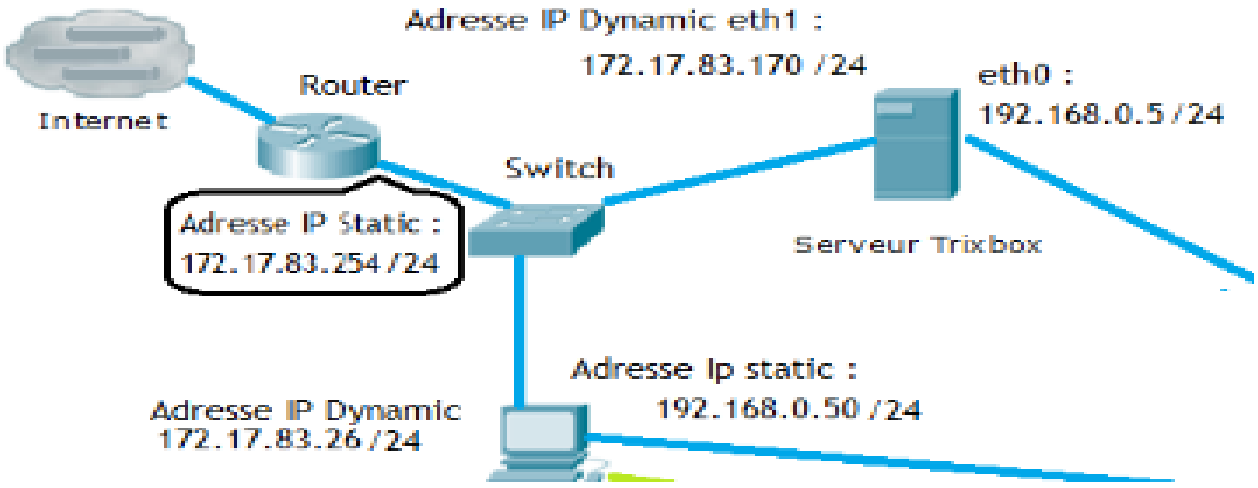
Nous nous situons ici comme testeur d'appel et BD.

7.1 Configuration Qt/MySQL

Pour rappel, l'IHM est programmée avec Qt, elle doit pouvoir accéder aux informations de la base de données MySql locale, ainsi que les modifier.

Afin de communiquer avec la base de données sous linux debian wheezy, le paquet « libqt4-sql-mysql » a du être installé via le gestionnaire de paquets « Synaptic ».

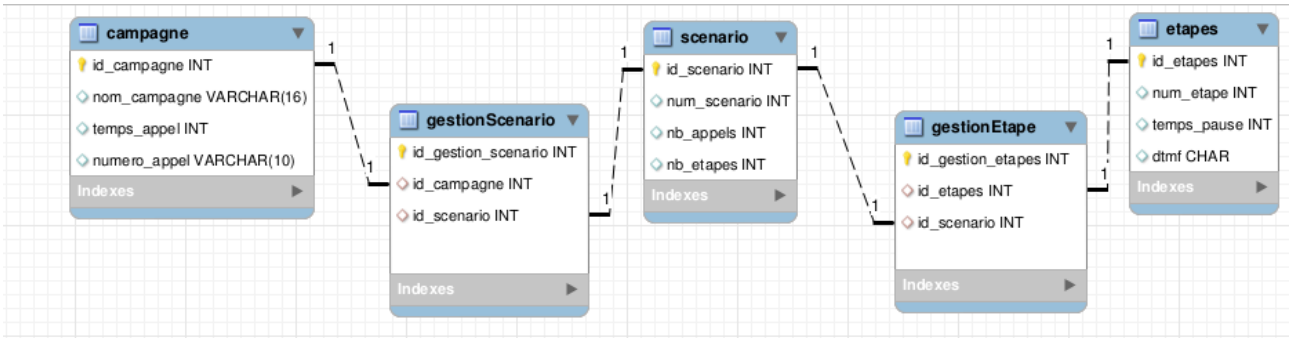
Sous QT, il faut aussi rajouter dans le .pro la ligne : QT += sql afin d'assurer la bonne liaison entre la base de données et l'application.



7.2 Principe d'accès à la base de données avec Qt4

Les principales fonctions de l'interface Homme-Machine de notre application, sont de permettre à l'utilisateur de paramétrer une nouvelle campagne, d'en charger/modifier une ancienne ou encore d'en supprimer une.

Voici les table de base de données que nous allons nous servir.



Il est donc nécessaire d'accéder à la base de données pour la consulter ou modifier du contenu des tables.

7.2.1 La connexion à la base de données:

Le framework Qt offre propose nativement des classes permettant l'accès et la manipulation du contenu des tables de base de données.

QSqlDriver est une classe abstraite permettant accéder à une base de données SQL.Il suffit de spécifier le driver nécessaire. Pour notre application, nous utiliserons QMYSQL.

Etant sous LINUX, il n'est pas nécessaire de recompiler le SDK de QT, car il intègre directement le drivers pour MYSQL, alors qu'il aurait fallut le faire si nous avions développé sous WINDOWS.

QSqlDriver ne peut être utilisé directement, nous avons besoin de QsqlDatabase.

QSqlDatabase est la classe qui représente la connexion à la base de données. Cette classe fournit une interface pour accéder à une base de données via une connexion. Une instance de QsqlDatabase, représentant la connexion, permet d'accéder à la base de données via l'un des pilotes de base de données pris en charge, qui sont issus de QsqlDriver.

Voici un exemple de code permettant de se connecter à la base de données :

```
Déclaration de la db :
private:
    QSqlDatabase db;
```

```
1.      db = QSqlDatabase::addDatabase("QMYSQL");
2.      db.setHostName("172.17.83.170");
3.      db.setPort(3306);
4.      db.setDatabaseName("MMAPondeuse");
5.      db.setUserName("root");
6.      db.setPassword("toto");
```

```
7.        db.open();
8.        bool verif = db.open();
9.        if ( verif ) qDebug() <<"Ouverture de la base de données avec succès";
10.       else qDebug() <<"Echec d'ouverture de la base de données";
```

Ligne 1: On spécifie que l'on va accèder au serveur de BD avec le driver QMYSQL

Ligne 2: On specifie que l'adresse de la base de données est 172.17.83.170.

Ligne 3: Port de connexion utilisé.

Ligne 4: Nom de la base de données

Ligne 5 : Login de la base de données.

Ligne 6: Le mot de passe d'accès à la base.

Ligne 7: Ouverture de la base de données.

Ligne 8-9-10: Le booléen verif, permet de connaître l'état de la connexion, un message dans le debugger s'affiche en fonction de cet état.

7.2.2 L'envoi d'une requête Sql préparée

La fonction prepare() prédispose la requête SQL à l'exécution.

Elle permet de construire des requêtes contenant des éléments variables.

Une seconde fonction, bindValue(), associe une valeur à un nom correspondant dans la requête SQL préparée. Ensuite, la fonction exec() exécute la requête Sql .

WindowTitle correspond au nom de la campagne de type Qstring, tempsAppelCampagne et numAppel sont des entiers et correspondent au temps d'appel de la campagne et au numéro d'appel.

Voici un exemple pour inséré le nom de la campagne, le temps de l'appel ainsi que le numéro d'appel dans la base de données.

```
1.  QSqlQuery query;
2.  query.prepare("insert into campagne (`nom_campagne`, `temps_appel`,
3.  `numero_appel`) values(:nomCampagne, :tempsAppelCampagne, :numAppel)");
4.  query.bindValue(":nomCampagne", windowTitle());
5.  query.bindValue(":tempsAppelCampagne",tempsAppelCampagne);
6.  query.bindValue(":numAppel",numAppel);
7.  query.exec();
```

Ligne 1: Déclaration de la requête.

Ligne 2-3: Préparation à l'envoi de la requête, la valeur à mettre à jour est celle précédée de :

Ligne 4-5-6 : Association des champs et des valeurs dans la requête.

Ligne 7: Envoi de la requête.

7.2.3 Gestion des erreurs d'envoi de la requête :

La fonction query.lastError() ; permet de vérifier si la dernière requête généré une erreur, ainsi si elle a retourné une valeur positive, un message d'erreur apparaît sur un QMessageBox l'indiquant à l'utilisateur.

Nous placerons donc ces lignes de codes après chaque requête SQL.

```
if (query.lastError().isValid())
  {
      QMessageBox::warning(this,"Erreur",query.lastError().text());
  }
```

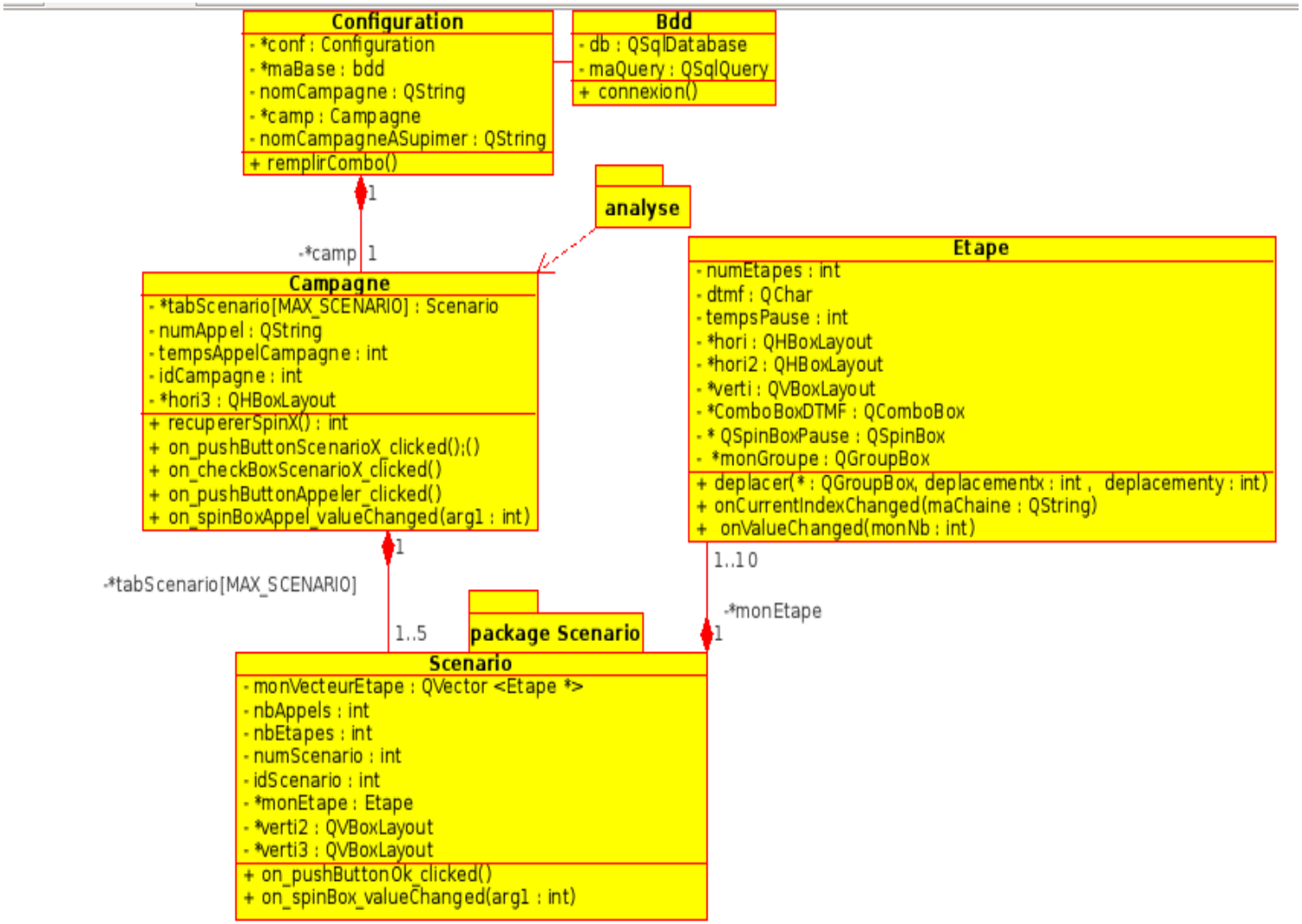
7.3 Fonctionnement de l'Interface Homme-Machine

Une interface Homme-Machine (IHM) a donc été créée à l'aide de QtCreator. Celle-ci doit permettre à l'utilisateur de configurer une nouvelle campagne, d'en lancer une ancienne ou bien d'en supprimer.

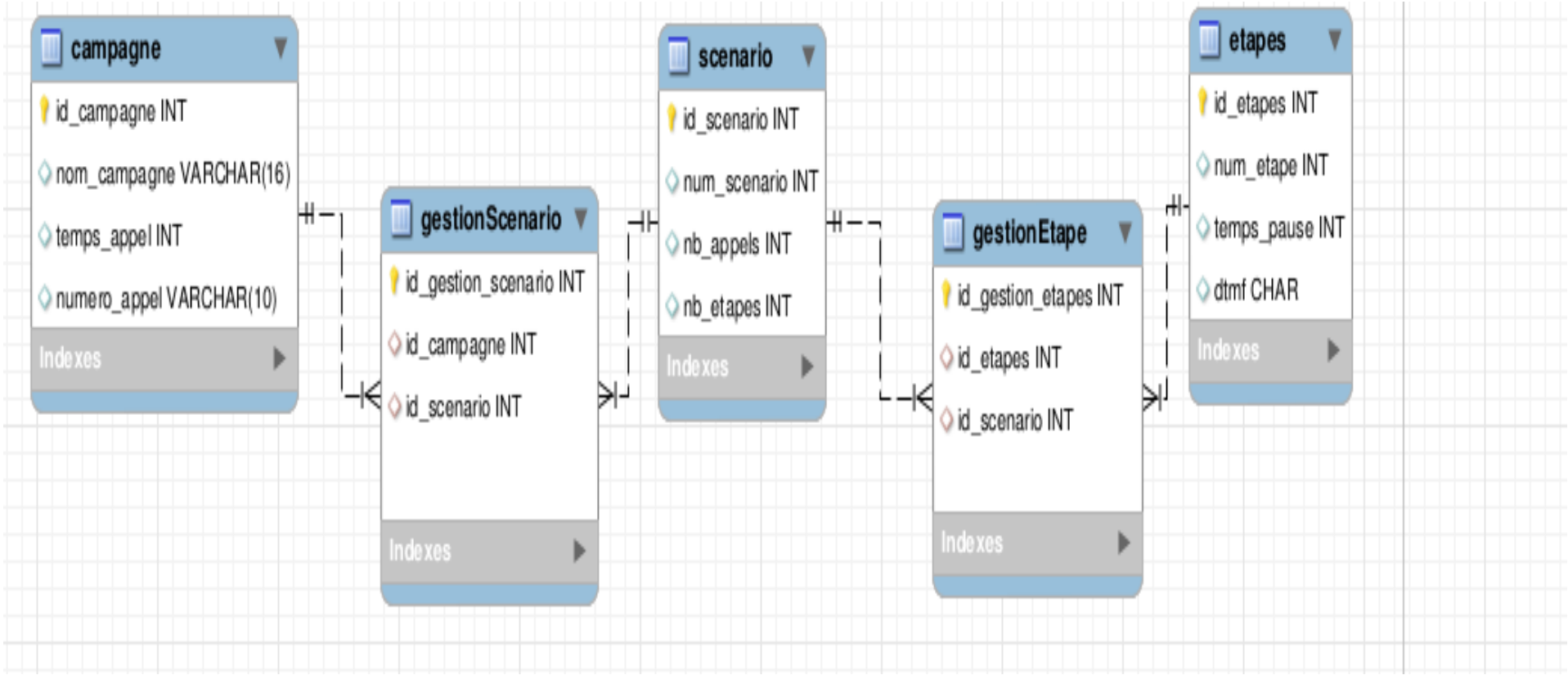
7.4 Nouvelle campagne, la configuration :

L'utilisateur peut avoir besoin de créer plusieurs campagnes différentes afin de tester le centre d'appel. Cette première fonctionnalité va le permettre.

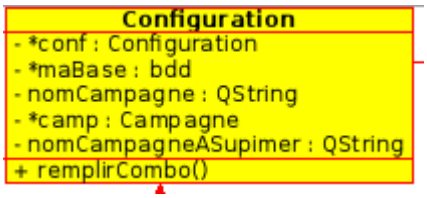
Classes utilisées :



Tables utilisées:



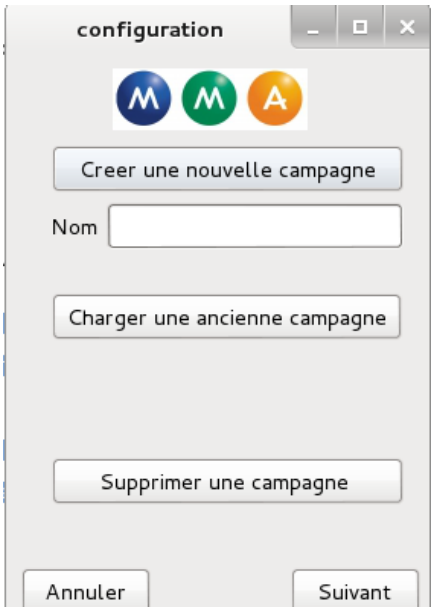
Tout d'abord l'utilisateur arrive sur l'IHM de configuration de campagne :



A l'apparition de l'IHM, nous la voyons composée de 3 boutons de type QpushButton offrant un choix à l'utilisateur (Nouvelle campagne, charger une ancienne campagne ou encore supprimer une ancienne campagne) ainsi que d'un QPushbutton annulation.

Afin d'éviter les mauvaises manipulation, les différents choix sont exclusifs.

Par exemple : lors de l'appui sur le bouton Nouvelle campagne, un QlineEdit apparaît pour laisser à l'utilisateur la possibilité d'entrer un nom de campagne, et masque toutes les autres fonctionnalités.



Pour toutes les parties de codes dans le dossier, les commentaires ont été supprimé pour une question de lisibilité, ils sont néanmoins présents dans le véritable code.

La creation d'une ihm avec QT creator genere la classe avec un attribut ui permettant d'accéder aux différent éléments placé via qtCreator

Voici le code correspondant au choix de création d'une nouvelle campagne.

Il est similaire pour les autres événements, sauf qu'il ne cache pas les mêmes fonctionnalités.

```
1. void Configuration::on_pushButtonCreerCampagne_clicked()
2. {
3.     ui->lineEditNomCampagne->show();
4.     ui->labelNomCampagne->show();
5.     ui->pushButtonSuivantConf->show() ;
6.     ui->comboBoxCampagne->hide() ;
7.     ui->comboBoxSupprimerCampagne->hide();
8.     ui->pushButtonSupprimer->hide();
9. }
```

Ligne 1: Définition du slot.

Ligne 3-4-5 : On affiche le lineEdit, le label et le button.

Ligne 6-7-8: On cache les comboBox et le buttonSupprimer.

La méthode show() ; permet d'afficher sur l'IHM, tandis que la méthode hide() ; permet de cacher sur l'IHM.

Voici la déclaration dans configuration.h de ce slot:

```
private slots:
void on_pushButtonCreerCampagne_clicked();
```

qt génère un fichier particulier il y a la connexion entre le signal click et le slot

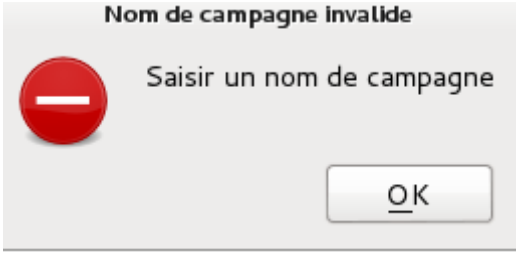
Le signal clicked() est émis lorsque l'utilisateur appuie sur le bouton nouvelle campagne.

Si le nom de campagne existe deja, l'utilisateur doit re-saisir un nom de campagne.

Dans le cas ou l'utilisateur crée une nouvelle campagne, un InputMask est mis en place dans la QlineEditCampagne de type : « nnnnnnnnnnnnnnn; », minimum 1 caractères, maximum 16.

Ce type de filtre va interdire à l'utilisateur de taper des noms de campagne trop long, ou avec des caractères spéciaux puisque le caractère n autorise uniquement des chiffres et des lettres.

Si l'utilisateur ne rentre pas de nom de campagne, un message d'erreurs apparaît, en revanche si le champ a été renseigné, on passe à l'IHM suivante :



Voici le code correspondant à l'appui sur le bouton suivant :

```
1. void Configuration::on_pushButtonSuivantConf_clicked()
2.  {
3.    nomCampagne= ui->lineEditNomCampagne->text();
4.
5.    if (nomCampagne=="")
6.      {
7.          QMessageBox::critical(0,tr("Nom de campagne invalide"), tr("Saisir
8.          un nom de campagne"));
9.       }
10.   else
11.      {
12.         camp->setWindowTitle(nomCampagne);
13.         camp->show();
14.         this->hide();
15.       }
16.  }
```

Ligne 1: Définition de la fonction.

Ligne 3: Récupération de la chaîne de caractères venant de la lineEdit de la saisi du nom de campagne.

Ligne 5-6-7-8:Vérification si la chaîne est NULL. Affichage d'un message d'erreurs.

Ligne 10: Si l'utilisateur a bien rentré une chaîne de caractères.

Ligne 11: Le titre de la fenêtre de la campagne est renommée en fonction de la chaîne rentrée par l'utilisateur.

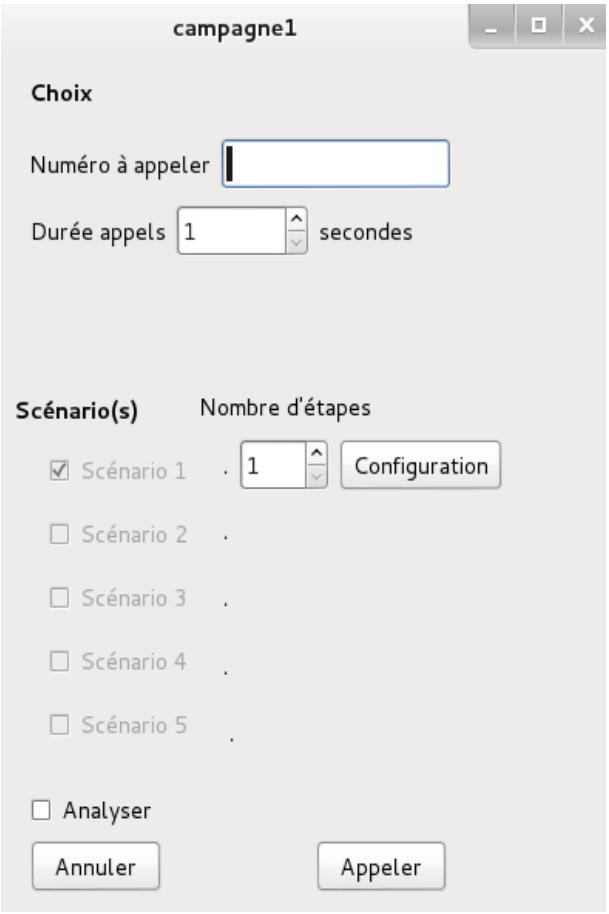
Ligne 12: Affichage de la fenêtre campagne.

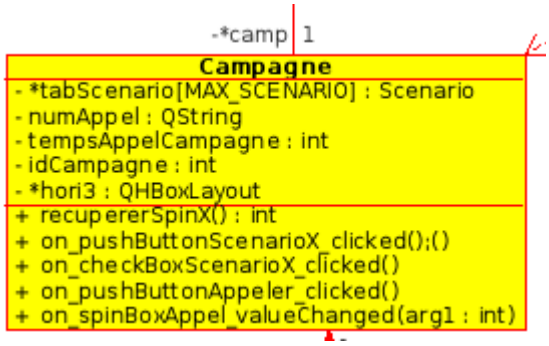
Ligne 13: La fenêtre de configuration est masquée.

7.4.1 Paramétrage de la campagne :

Nous arrivons alors sur une autre IHM : la campagne

Cette IHM va permettre à l'utilisateur de choisir le numéro à appeler, le temps de l'appel, le nombre de scénarios souhaité ainsi que le nombre d'étapes par scénario.





La valeur de la QlineEdit: lineNumAppel permet à l'utilisateur d'entrer le numéro vers lequel les appels seront effectués. Elle dispose d'un InputMask de type « 9999999999 » qui oblige la saisie uniquement de 10 chiffres maximum puis stockée dans l'attribut numAppel de la classe Campagne.

En cas de champ vide, un message d'erreur apparaît.

La valeur est récupérée lors de l'appel à la méthode

Campagne::on_pushButtonAppeler_clicked() ;

(lorsque l'utilisateur appuie sur le bouton appeler.)

Voici le code correspondant:

```
1.     numAppel = ui->lineEditNumAppel->text();
2.     if (numAppel=="")
3.     {
4.         QMessageBox::critical(0,tr("Numero d'appels invalide"), tr("Saisir un
5.         numero d'appels"));
6.     }
```

Ligne 1: Récupération de la valeur du numéro d'appel dans numAppel.

Ligne 2-3-4-5-6:Si le champ est vide alors un message d'erreurs apparaît.

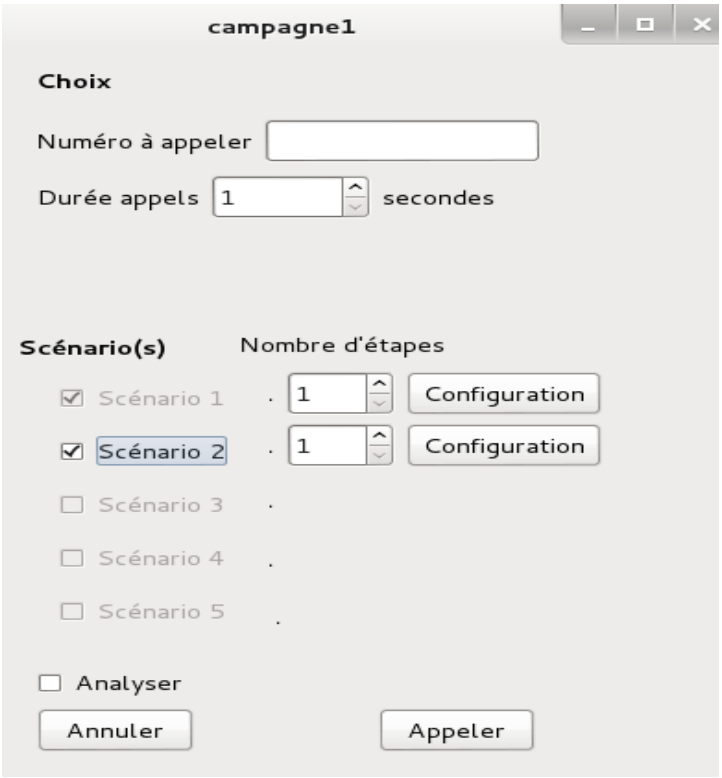
La valeur de la QspinBox: spinBoxTempsAppel est récupérée grâce au slot, puis stockée dans l'attribut tempsAppelCampagne de la classe campagne, voici le code correspondant :

```
1. void Campagne::on_spinBoxAppel_valueChanged(int arg1)
2.     {
3.           tempsAppelCampagne=arg1;
4.     }
```

Ligne 1: Définition de la fonction, arg1 posséde donc la valeur du spinBox.

Ligne 2-3-4: tempsAppelCampagne prend la valeur de arg1.

Passons maintenant à la gestion des scénarios. Pour une campagne, nous avons besoin d'au minimum un scénario, cependant il a fallu coder l'interface de manière à obliger l'utilisateur à paramétrer d'abord le scénario 1 ,puis le 2 … (en cas de plusieurs scénarios) afin que l'utilisateur ne paramètre pas le scénario 2 avant le scénario 1.



La checkbox du scénario sera toujours checked sans possibilité de modifier la valeur, cependant pour les 4 autres scénarios, les checkBox, spinBox, et pushButton ne sont pas disponibles ou visibles.

Hide pour cacher l'élément.

setEnabled(false) pour rendre l'élément indisponible.

```
    ui->pushButtonScenario2->hide()
    ui->spinBoxScenario2->hide();
    ui->checkBoxScenario2->setEnabled(false);
```

Nous avons ainsi un code similaire pour les autres scénarios.

7.4.2 Paramétrage d'un scénario :

Pour un scénario nous récupérons la valeur du spinBox pour connaître le nombre d'étapes saisis par l'utilisateur associées au scénario.

Pour le scénario 1:

```
1. int Campagne::recupererSpin1()
2.          {
3.    int nbEtapesSc1;
4.    nbEtapesSc1=ui->spinBoxScenario1->text().toInt();
5.    return nbEtapesSc1;
6.          }
```

Ligne 1: Définition de la fonction.

Ligne 2-3-4-5-6: on récupère le nombre étapes et on convertit en entier grâce a la fonction toInt() ; on retourne le nombre d'étapes.

Une fois que l'utilisateur a appuyé sur le bouton de scénario 1 on créé un nouveau scénario en créant une instance de la classe Scenario que l'on place dans le tableau de scénario.

Il n'est possible de créer autre scénario, que si le scénario précédent est validé.

Voici le code correspondant pour le scénario 1 :

```
1. void Campagne::on_pushButtonScenario1_clicked()
2.  {
3.    int nbEtapesSc1;
4.    nbEtapesSc1=recupererSpin1();
5.    Scenario *sc1 = new Scenario(nbEtapesSc1,1);
6.    tabScenario[0]=sc1
7.    ui->checkBoxScenario2->setEnabled(true);
8.  }
```

Ligne 3-4: On récupère le nombre d'étapes.

Ligne 5: On crée un nouveau scénario sc1.

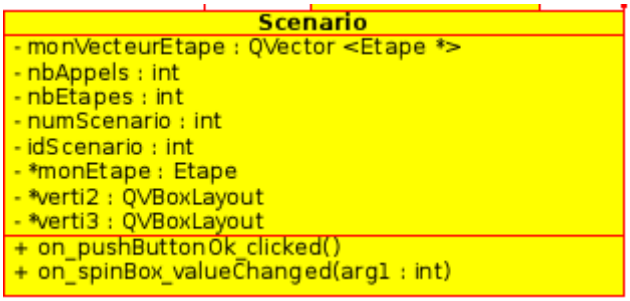
Ligne 6: On ajoute ce scénario dans la première case du tableau.

Ligne 7: On rend cochable la checkboxscenario2 permettant d'ajouter un second scénario.

Le code est similaire pour les scénarios suivants.

Les données de la campagne étant maintenant définies, nous allons aborder les scénarii.

Voici la classe Scenario :



Le constructeur de Scenario prend en paramètre le nombre d'étapes(nbEtapes) et le numéro de scénario(numScenario).

Le code qui permet de renommer le titre de l'IHM et le nom de la fenêtre est le suivant.

```
1.  Scenario::Scenario(int nb_Etapes, int
2.  num_Scenario):nbEtapes(nb_Etapes),numScenario(num_Scenario),
3.  QWidget(0),
4.  ui(new Ui::Scenario)
5.  {
6.    QString nomScenario;
7.    QString nomFen;
8.    nomScenario.sprintf("Scenario %d",numScenario);
9.    this->setWindowTitle(nomScenario);
10.   nomFen.sprintf("Scenario%d",numScenario);
11.   this->setObjectName(nomFen);
```

On déclare 2 variables de types Qstring puis on fait appel a la fonction sprint.Puis setWindowsTitle et setObjectname pour assigner le titre et le nom de la fenêtre.

Nous avons besoin de créer différents layout pour disposer nos étapes qui seront crées dynamiquement, ce sont des attributs de la classe scenario:

```
hori3=new QHBoxLayout(this);
verti2 =new QVBoxLayout(this);
verti3=new QVBoxLayout(this);
```

Un horizontal et 2 vertical.

Redimensionnement de la fenêtre :

Il faut redimensionner la fenêtre afin d'afficher correctement toutes les étapes, si il y a plus que 6 étapes on déplace les 5 suivantes sur l'axe X.

```
1. int taillex,tailley, value;
2. if(nbEtapes < 6)
3.    {
4.     taillex = 400;
5.     value = nbEtapes;
6.    }
7.   else
8.     {
9.   taillex = 800;
10.  value = 5;
11.    }
12. tailley = value *140 + 130;
13. this->resize(taillex,tailley);
```

Ligne 1:Taille x et y de la fenêtre, value est le coefficient multiplicateur en fonction du nombre d'étape.

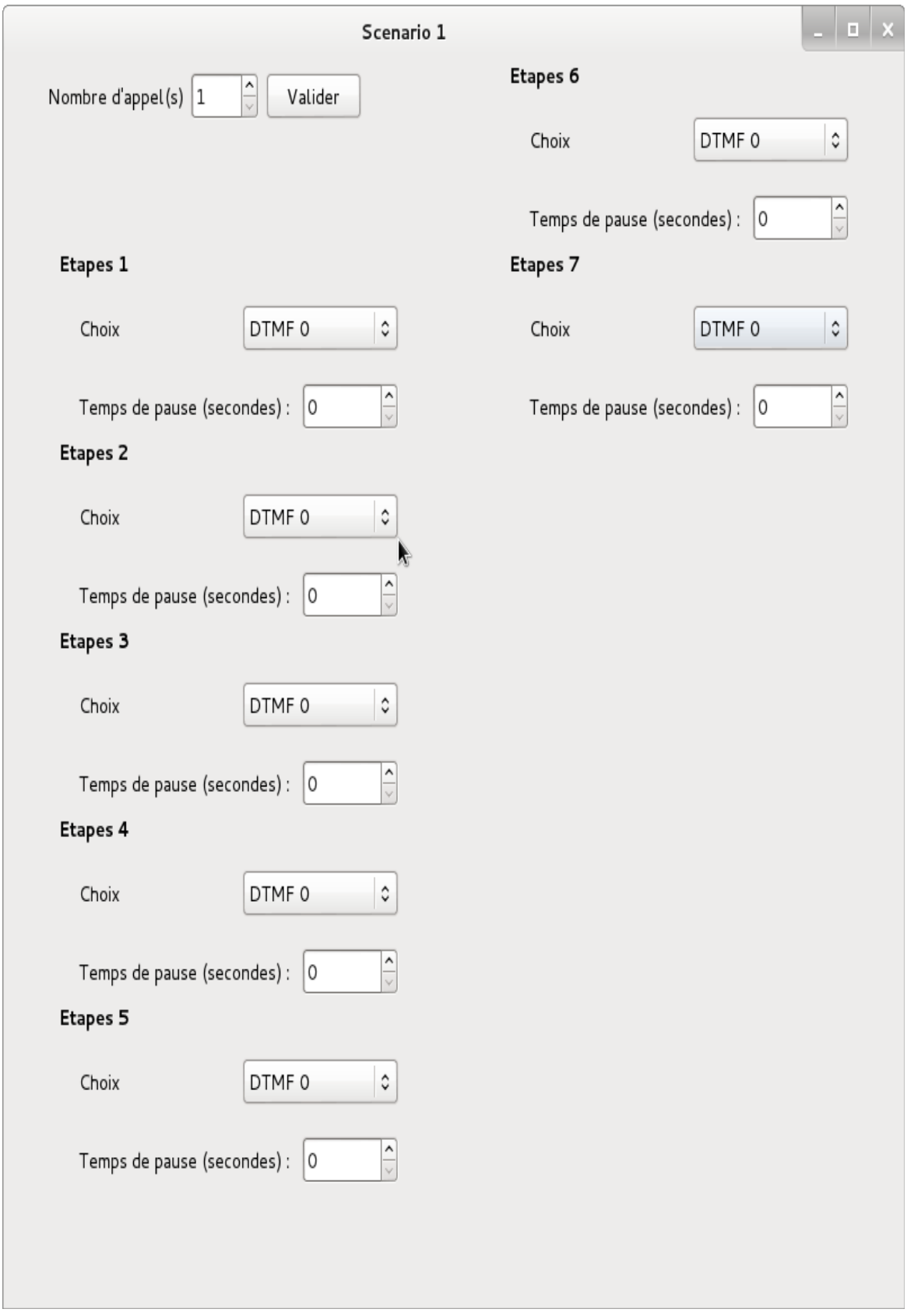
Ligne 2-3-4-5-6: Si il y a moins de 6 étapes, l'axe X de l'IHM mesure 400px

Ligne 7-8-9-10-11 : Sinon on double la taille de l'IHM sur l'axe X.

Ligne 12: tailley augmente en fonction du nombre d'étapes.

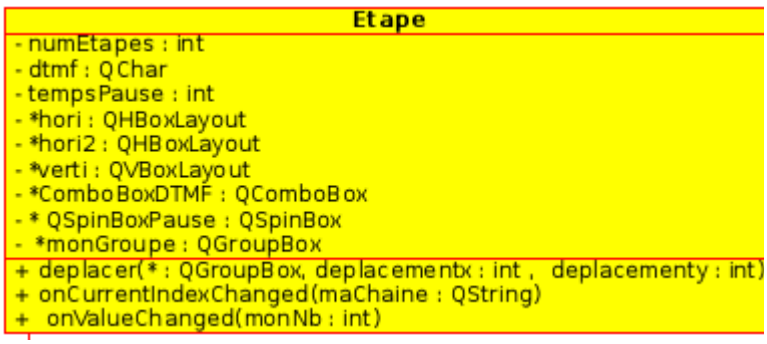
Ligne 13: Redimentionnement de la fenêtre.

Voici ce que cela donne avec 7 étapes :



On instancie autant d'objet de type Etape que d'étapes définies dans le scénario.

Voici la classe Etape :


Etape
- numEtapes : int
- dtmf : QChar
- tempsPause : int
- *hori : QHBoxLayout
- *hori2 : QHBoxLayout
- *verti : QVBoxLayout
- *ComboBoxDTMF : QComboBox
- * QSpinBoxPause : QSpinBox
- *monGroupe : QGroupBox
+ deplacer(* : QGroupBox, deplacementx : int , deplacementy : int)
+ onCurrentIndexChanged(maChaine : QString)
+ onValueChanged(monNb : int)

monEtape la classe Scenario (Etape *monEtape;). L'étape créée est placée dans le layout vertical, on les place ensuite dans le vecteur Etapes déclaré dans scenario.h (QVector <Etape *>monVecteurEtape;)

Grâce a la fonction push_back qui place l'étape a la fin du vecteur puis on affiche fenêtre:

```
1.     for(int i = 1; i <= nbEtapes; i++)
2.             {
3.                  monEtape =new Etape (this, i);
4.                  verti2->addWidget(monEtape);
5.                  monVecteurEtape.push_back(monEtape);
6.              }
```

Ligne 1: Pour le nombres d'étapes choisis par l'utilisateur.
Ligne 3: On créé une nouvelle étape de type Etape.

Ligne 4: On ajoute l'étape au layout vertical.
Ligne 5: On met l'étape dans le vecteur.

La classe Etape :

L'étape est créée à partir du constructeur : Etape::Etape(QWidget* widgetsc, int numEtapes)

Une étape est mise dans un QgroupBox, elle possède un numéro (de 1 à 10). A l'intérieur du comboBox nous créons un nouveau comboBox dans lequel nous insérons les valeur des DTMF(de 0 à 9 plus # et *) grâce à la fonction insertItems() ;

```
1. monGroupe=new QGroupBox(widgetsc);
2. QString nomEtape;
3. nomEtape.sprintf("Etapes %d",numEtapes);
4. monGroupe->setTitle(nomEtape);
5. ComboBoxDTMF=new QcomboBox(monGroupe);
6. ComboBoxDTMF->clear();
7. ComboBoxDTMF->insertItems(0, QStringList() <<
8. QApplication::translate("Scenario1", "DTMF 0", 0,
9.    QApplication::UnicodeUTF8));
```

Ligne 1: Création du nouveau QgroupBox.

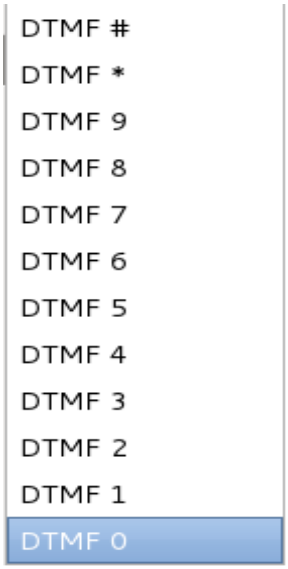
Ligne 2: Création de la variable nomEtape.

Ligne 3-4: On renomme le groupBox avec le numéro d'étapes associées.

Ligne 5: Création du nouveau QcomboBox.

Ligne 6 : Initialisation du comboBox.

Ligne 7-8-9: Insertion des différents DTMF dans le comboBox (ici seulement le DTMF0 est montré)



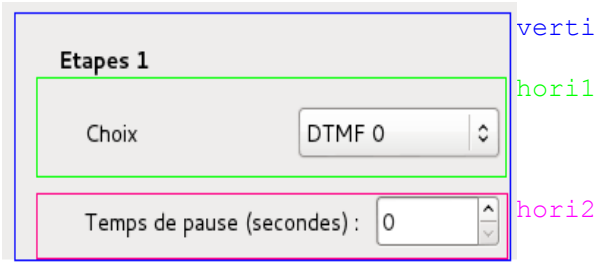
Nous avons aussi besoin d'une QspinBox pour le temps de pause, cette dernière est limitée de 0 à 59 secondes :

```
QSpinBoxPause = new QspinBox(monGroupe) ;
QSpinBoxPause->setMaximum(59);
QSpinBoxPause->setMinimum(0);
```

Voici le code permettant de positionner les différents éléments dans les layout

le schéma et le code correspondant.

La fonction setFocus() ; redonne le focus à l'utilisateur, lui permettant d'avoir accès aux différentes étapes.




```
1. hori = new QHBoxLayout();
2. hori2 =new QHBoxLayout();
3. verti = new QVBoxLayout(monGroupe);
```

Ligne 1-3-3 :Création des layout

```
1.   hori->addWidget(new QLabel("Choix"));
```

```
2.    hori->addWidget(ComboBoxDTMF);
3.    hori2->addWidget(new QLabel("Temps de pause (secondes) : "));
4.    hori2->addWidget(QSpinBoxPause);
5.    verti->addLayout(hori);
6.    verti->addLayout(hori2);
```

Ligne 1-2: On met les éléments dans le layout Hori.

Ligne 3-4: On met les éléments dans le layout Hori2.

Ligne 5-6: On met les horizontal dans le layout vertical Verti.

```
1.   QSpinBoxPause->setFocus();
2.   ComboBoxDTMF->setFocus();
```

Ligne 1-2: On redonne le focus.

Le focus sert a redonner la possibilité d'interagir avec les composant grace au curseur de la souris.La fonction deplacer() ; permet de déplacer les étapes afin qu'elles ne se superposent pas sur l'IHM.

```
1. #define DEPLACEMENT_Y 125
2. #define DEPLACEMENT_X 50
3. #define DEPLACEMENT_X1 450
4. void Etape::deplacer(QGroupBox* monGroupe,int deplacementx,int
5. deplacementy)//Méthode permetant de déplacer les étapes sur l'ihm
6.  {
7.    monGroupe->setGeometry(deplacementx,deplacementy,310,150);
8.  }
```

Ligne 1-2-3: Déclaration des constantes.

Ligne 4-5: Déclaration de la fonction deplacer.

Ligne 7: Déplacement du groupe.

```
1. if(numEtapes < 6)
2.   {
3.         deplacer(monGroupe, DEPLACEMENT_X,(numEtapes)*DEPLACEMENT_Y );
4.          }
5. else
6.          {
7.          deplacer(monGroupe, DEPLACEMENT_X1 ,(numEtapes-6)*DEPLACEMENT_Y );
8.          }
```

Ligne 1-2-3-4: On déplace les 5 premieres étapes en Y.
Ligne 5-6-7-8: Si il y a plus que 5 étapes, on les déplacent sur l'axe X.

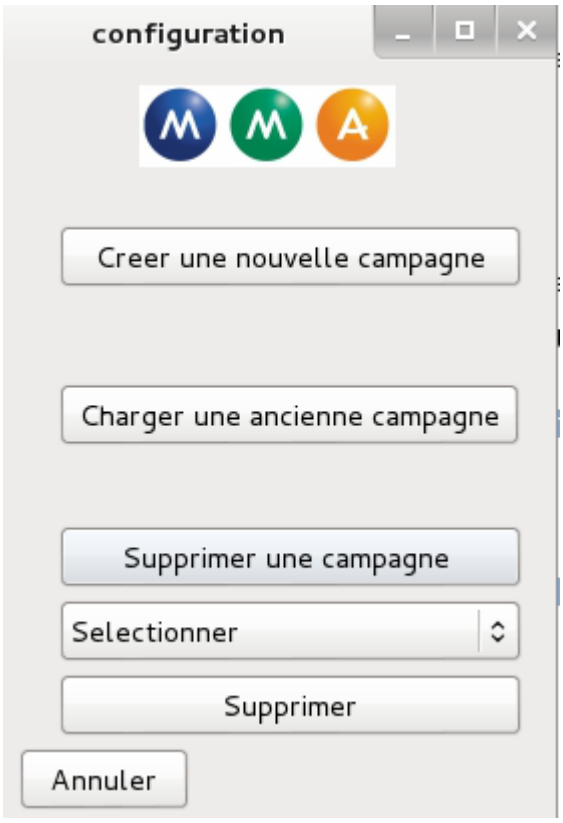
```
1.connect(ComboBoxDTMF,SIGNAL(currentIndexChanged(QString)),this,SLOT(onCurrentI
2.ndexChanged(QString)));
3.connect(QSpinBoxPause,SIGNAL(valueChanged(int)),this,SLOT(onValueChanged(int))
```

```
4.);
5. void Etape::onCurrentIndexChanged(QString dtmfChaine)
6.   {
7.            dtmf=dtmfChaine.at(dtmfChaine.length()-1);
8.   {
9. void Etape::onValueChanged(int leTempsDePause)
10.  {
11.         tempsPause=leTempsDePause;
12.  }
```

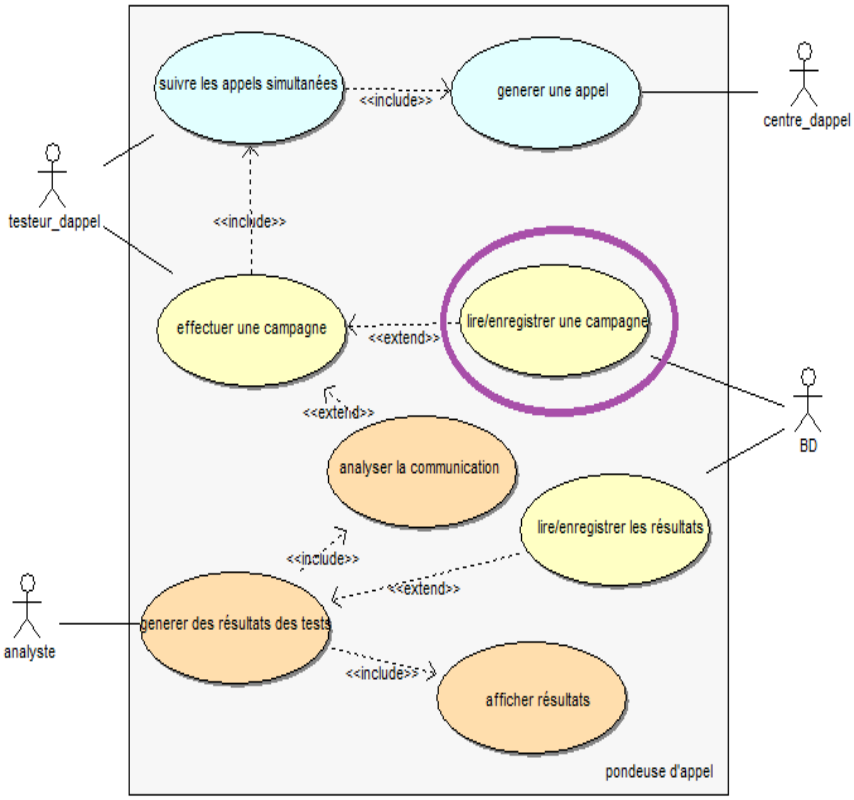
Ligne 1-2 5-8: On récupère la valeur du comboBox.

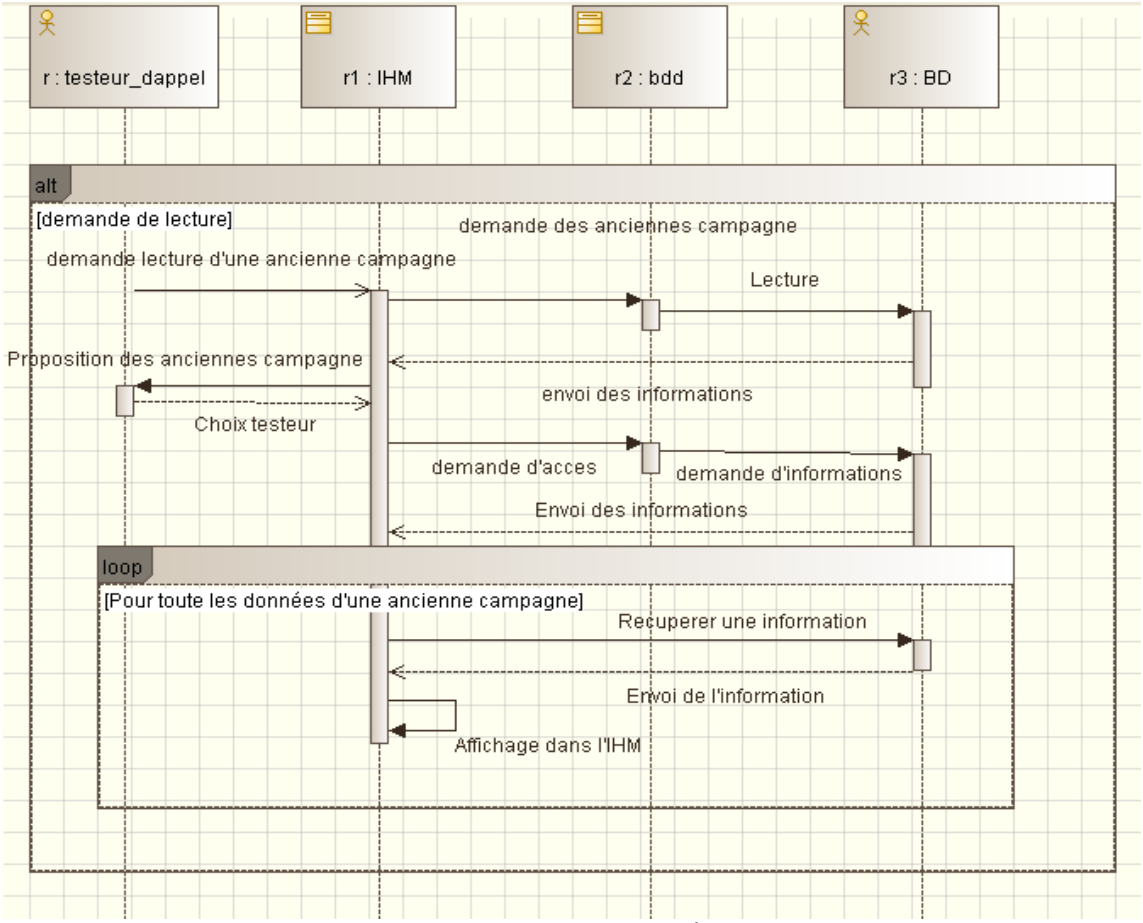
Ligne 2-4 9-12: On récupère la valeur du SpinBox.

Lors de l'affichage des anciennes campagnes, les comboBoxCampagne et comboBoxSupprimer campagne sont remplis grâce aux informations de la base de données.



Nous ferons donc appel au cas d'utilisation lire une campagne :





Voici le code permettant de remplir les comboBox grâce au nom de campagnes enregistrées dans la base de données :

```
1.  void Configuration::remplirCombo()
2.   {
3.     QSqlQuery query;
4.     query.prepare("SELECT nom_campagne FROM campagne");
5.     query.exec() ;
6.     while(query.next())
7.           {
8.     ui->comboBoxCampagne->addItem(query.value(0).toString());
9.     ui->comboBoxSupprimerCampagne->addItem(query.value(0).toString());
10.          }
11.    if (query.lastError().isValid())
12.             QmessageBox::warning(this,"Erreur",query.lastError().text());
13.    query.finish();//fin de la requete

       }
```

Ligne 1: Définition de la fonction.

Ligne 2: Début de la fonction.

Ligne 3: Création de la query de type QsqlQuery.

Ligne 4: Compilation de la requête.

Ligne 5: Exécution de la requête.

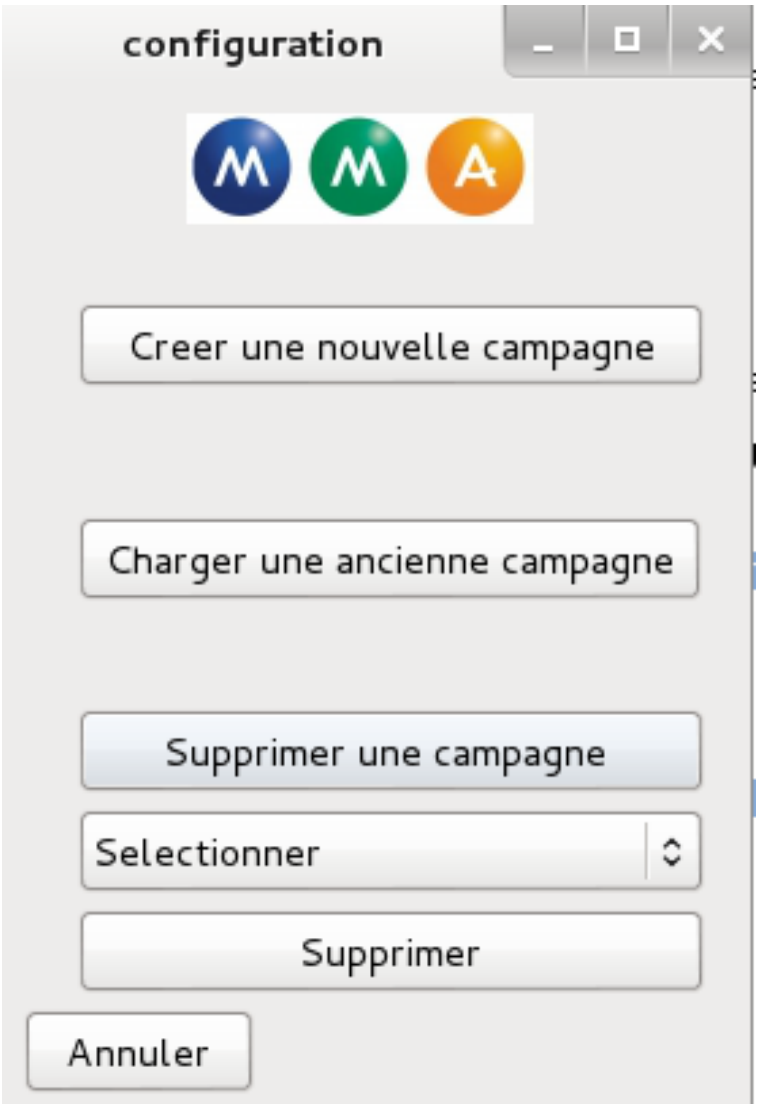
Ligne 6-10: On sélectionne toutes les campagnes de la base de données, puis on les ajoutes dans les comboBox, la fonction toString permet de convertir en chaîne de caractères.

Ligne 11-12 : Traitement des erreurs.

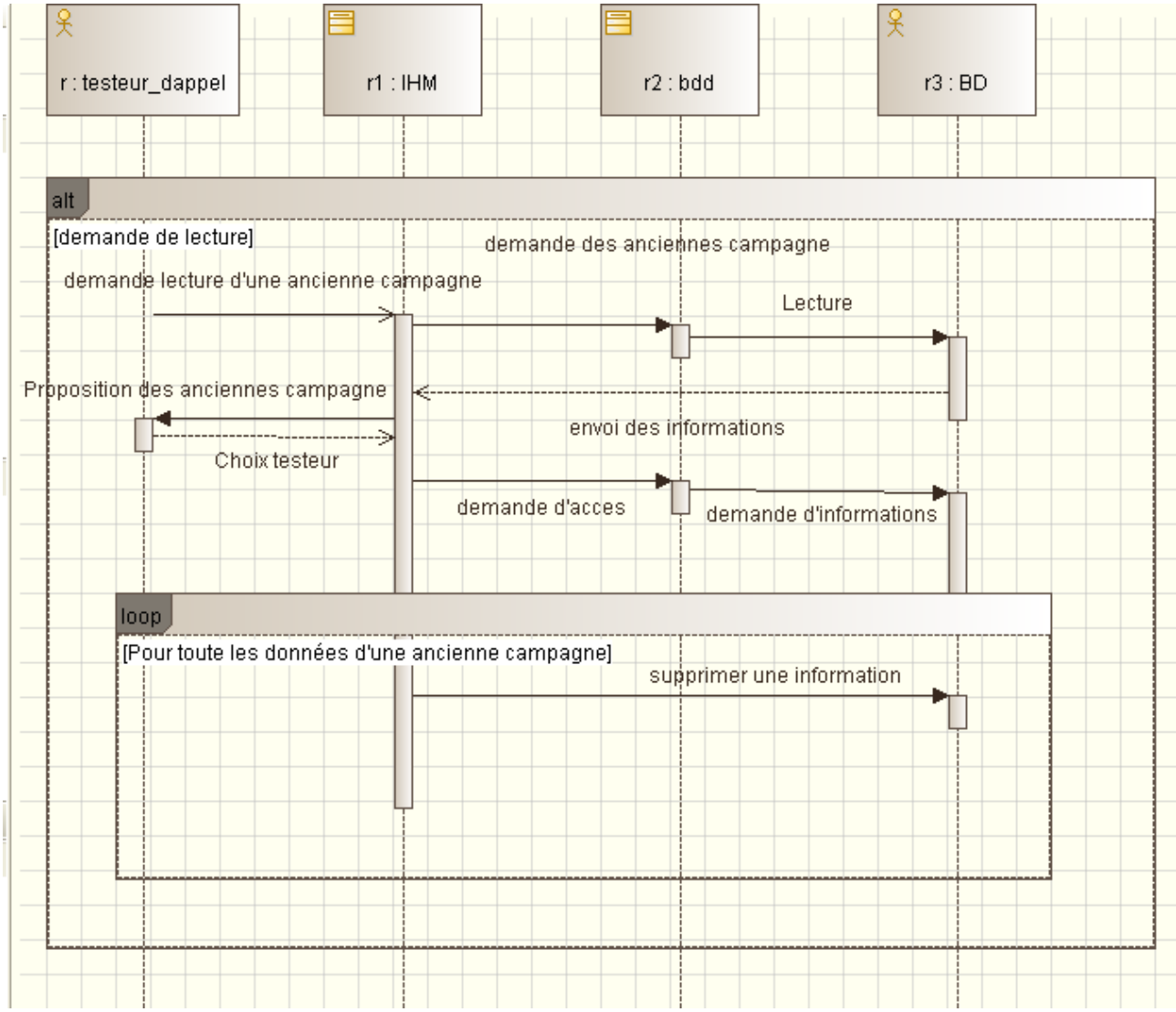
Ligne 13: Fin de la query.

7.4.3 Supprimer une campagne :

Si l'utilisateur veut supprimer une campagne de la base de données, il le fera à partir de l'IHM de configuration, après un appui sur le bouton Supprimer une campagne.



Voici le diagramme de séquence :




```
1. void Configuration::on_pushButtonSupprimer_clicked()
2.     {
3.         QSqlQuery query;
4.         query.prepare("DELETE FROM `campagne` WHERE `campagne`.`nom_campagne`
5.         =(:nomCampagneASupprimer)");
6.         query.bindValue(":aSupprimer",nomCampagneASupprimer);
7.         query.exec();
8.         query.finish();//fin de la requête
9.     }
```

Ligne 1: Définition de la fonction.

Ligne 2: Début de la fonction.

Ligne 3: Création de la query de type QsqlQuery.

Ligne 4-5: Compilation de la requête.

Ligne 6: Exécution de la requête.

Ligne 7: Association des champs aSuprimer et nomCampagneASuprimer

Ligne 8: Fin de la query.

La suppression d'une campagne entraîne la suppression des scénarios qui entraîne la suppression des étapes.

L'ordre de la suppression est inversé il faut d'abord supprimer les étapes puis les scénarios puis les campagnes.

7.5 Conclusion :

Par rapport aux différentes fonctionnalités dont j'avais la charge, presque toutes sont fonctionnelles.

A savoir :

-La création d'une campagne.

-L'enregistrement d'une campagne dans la base de données.

-La suppression d'une campagne.

Lors de la rédaction de ce dossier, quelques parties du projet n'ont pas encore été traitées, notamment la partie qui permet de charger les étapes d'une ancienne campagne à partir de la base de données. En effet, le temps prévisionnel n'a pas été respecté, plusieurs IHM de test ont été réalisés , puis abandonnés pour enfin avoir un résultat final, valide par rapport au demandeur.

La mise en place des étapes générées dynamiquement via les données de la campagne a pris plus de temps que prévu.

Plusieurs bases de données ont aussi été créées, puis abandonnées, avant d'arriver aux tables de la base de données actuelle.

Le travail de codage de longue haleine a donc été très intéressant, et m'a permis de me perfectionner dans le langage C++. La bibliothèque QT m'a simplifié le travail en liant l'aspect graphique et codage.

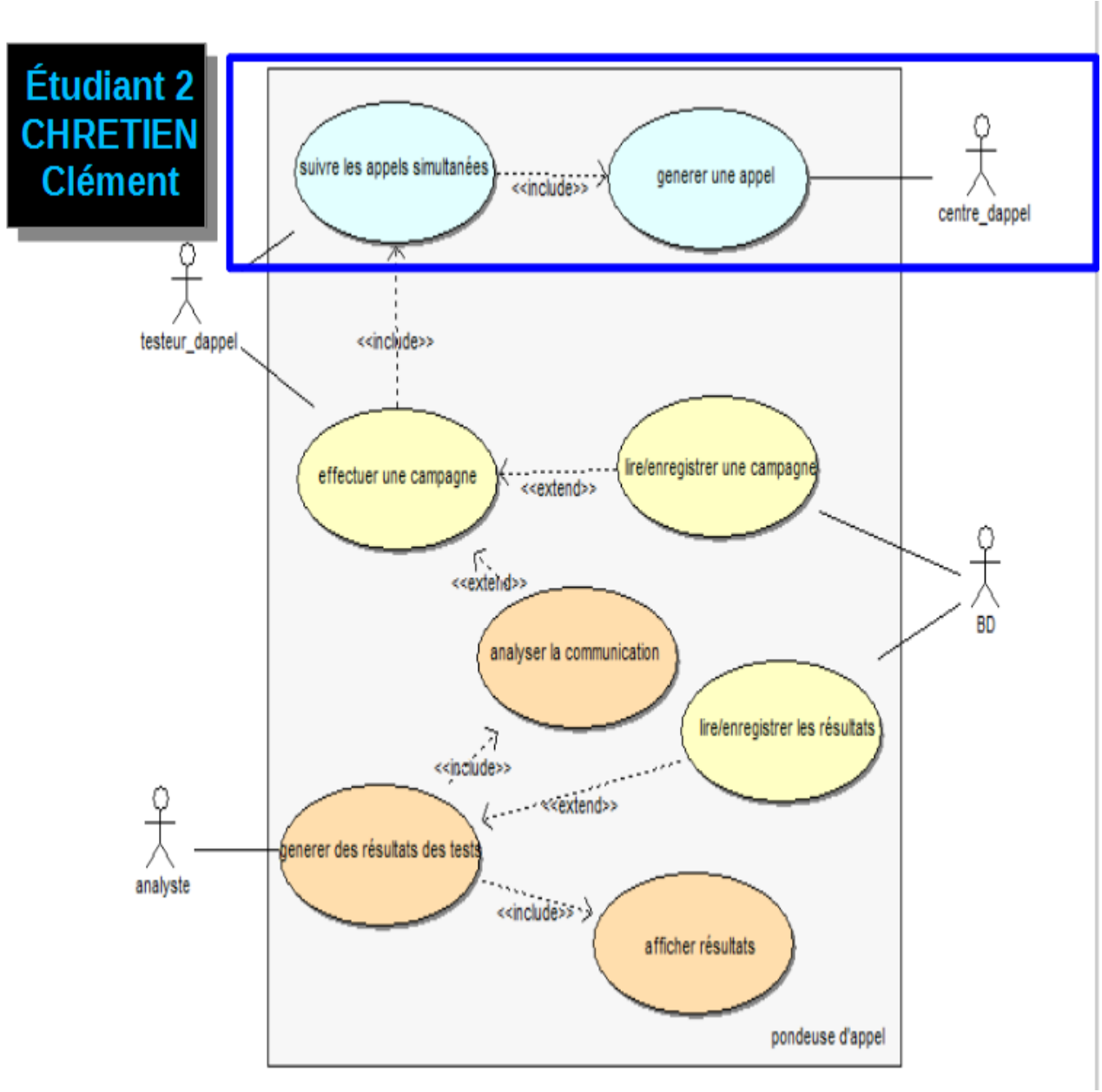
Bien que que mon avenir ne soit pas totalement arrêté, je me sens aujourd'hui plus orienté vers les réseaux plutôt que vers la programmation/développement. Cependant, même si ce projet représentait une partie importante de programmation , je ne l'ai en aucun cas, trouvé ennuyeux.

Les contraintes posées par l'entreprise m'ont aidé tout au long du projet a ne pas sortir du sujet.

Malgré des écarts ressentis sur l'avancée du travail, la dynamique de groupe a permis de faire avancer le projet sur les différentes parties.

Pour ma part, ce projet reste une très bonne expérience.

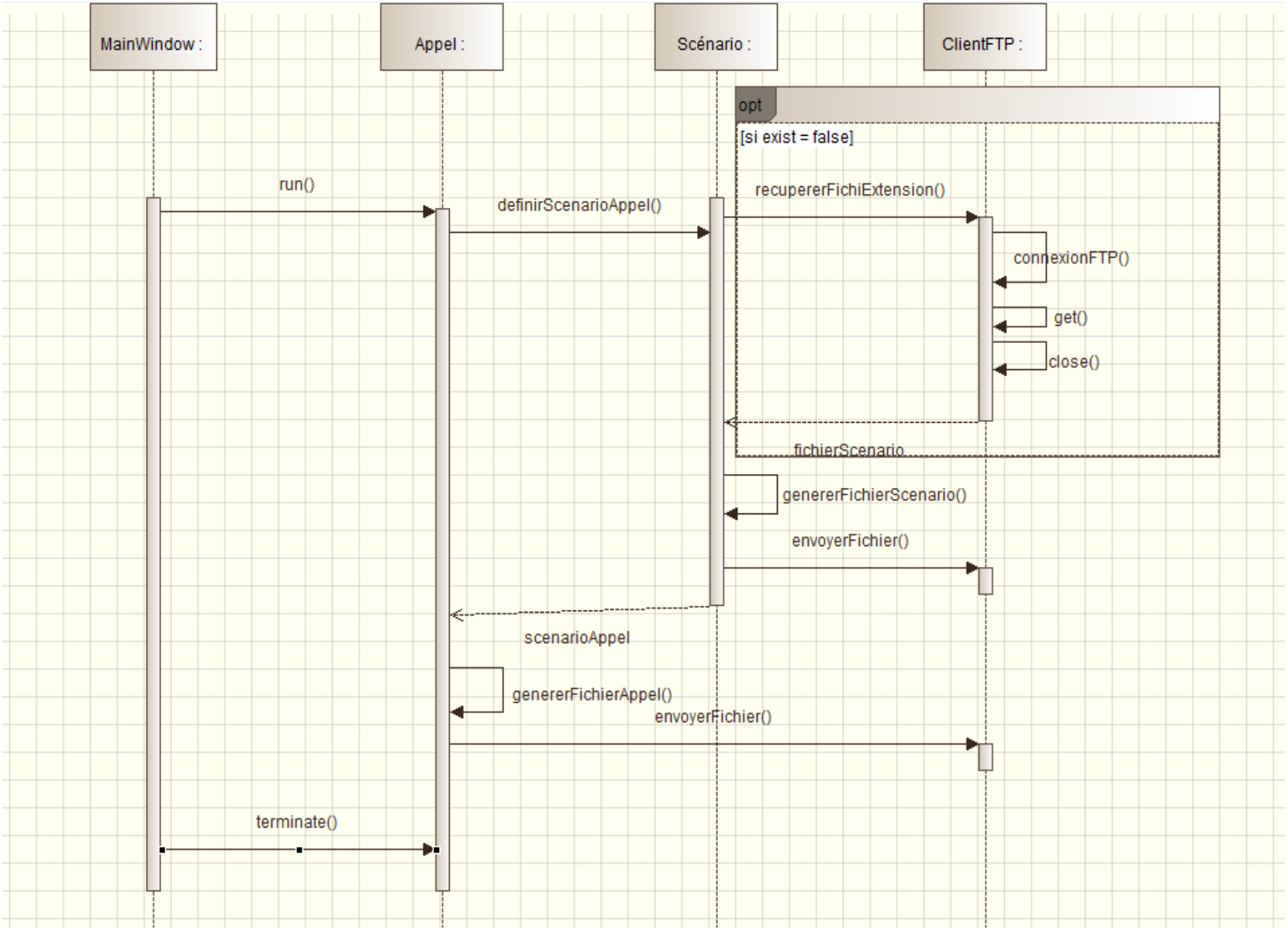
8. Conception détaillée : Partie Chrétien Clément (E2)



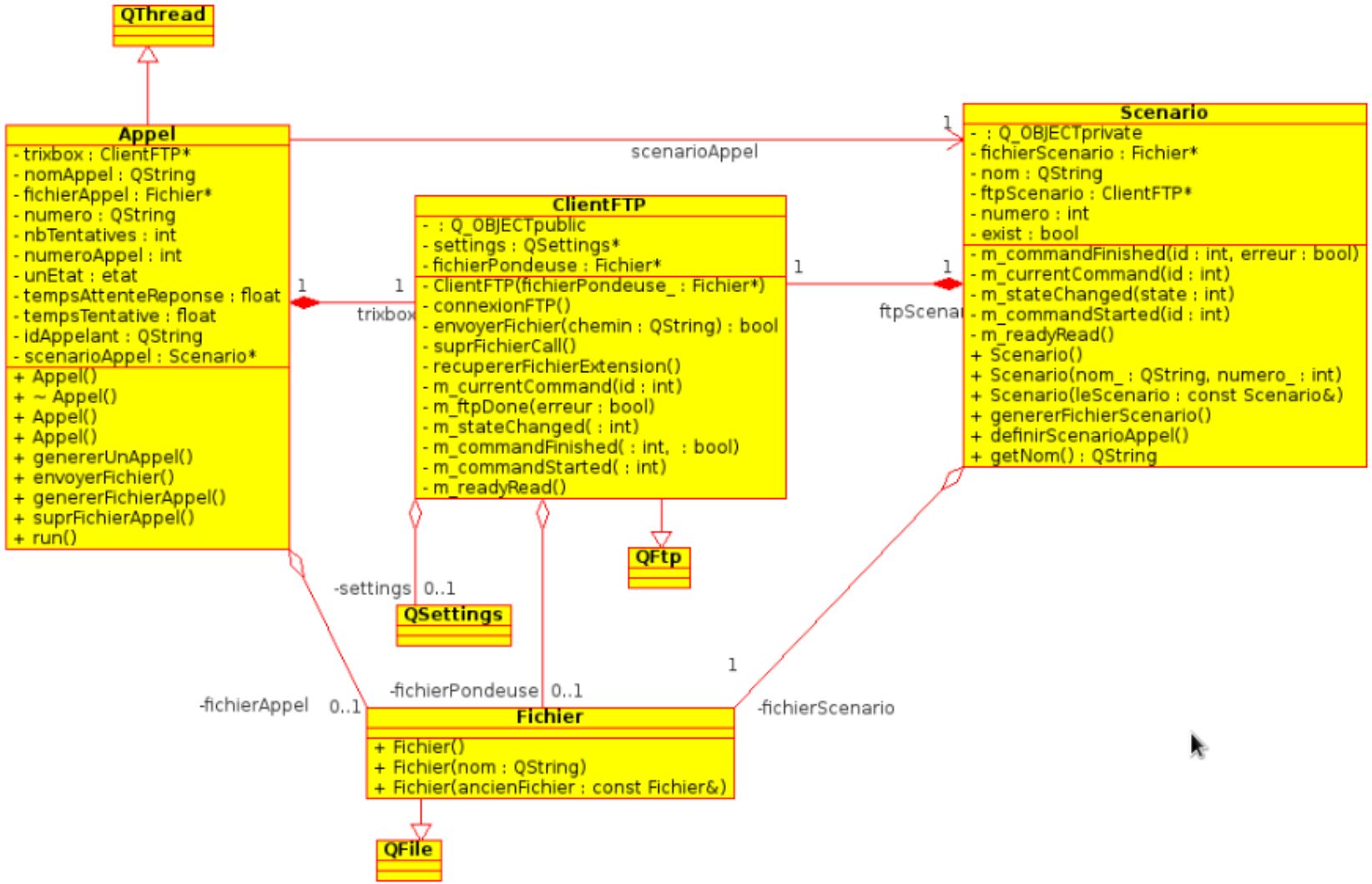
Dans le cadre bleu se situent les cas d'utilisations qui concernent ma partie et que je dois réaliser

Voici un schéma présentant le transit des différents fichiers de l'application entre le PC Client et la TRIXBOX :

8.1 Diagramme de Séquence pour la génération d'un appel :

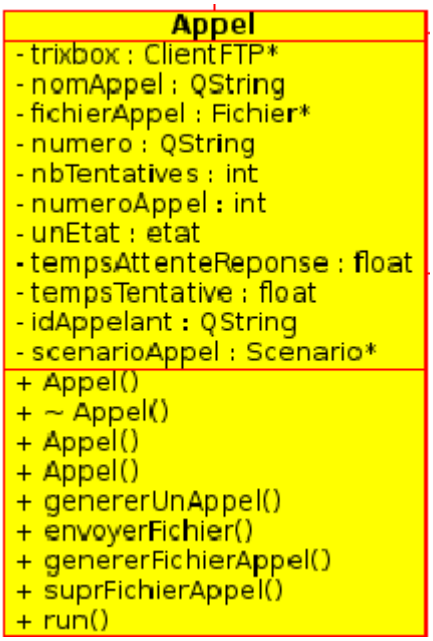

MainWindow :
Appel :
Scénario :
ClientFTP :
opt
[si exist = false]
run()
definirScenarioAppel()
recupererFichiExtension()
connexionFTP()
get()
close()
fichierScenario
genererFichierScenario()
envoyerFichier()
scenarioAppel
genererFichierAppel()
envoyerFichier()
terminate()

8.2 Diagramme de Classes



Voici les modifications qui ont été apportées au diagramme de classe initialement prévu :

La classe Appel :

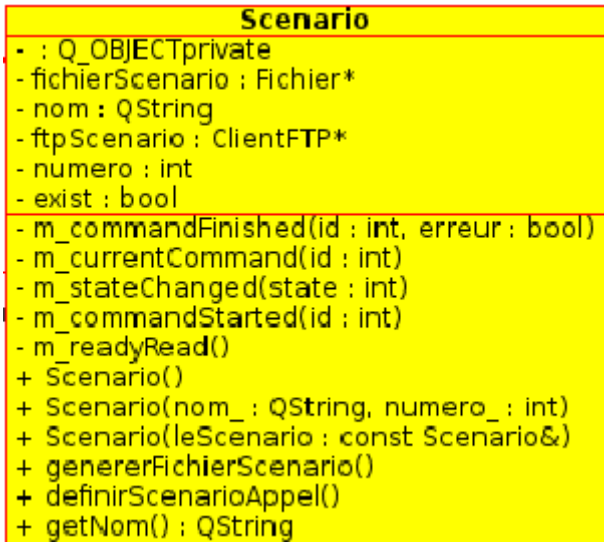


- Attributs :
  - numeroAppel : Numéro à appeler
  - numero : numéro de l'appel
  - tempsAttenteReponse : temps d'attente avant que l'appel ne soit considéré comme avorté
  - tempsTentative : temps d'attente avant la prochaine tentative de l'appel
  - nbTentative : nombre de tentatives
  - idAppelant: nom qui s'affichera sur le téléphone
  - trixbox : instance de la classe ClientFTP qui fait transité les fichier .call vers la TRIXBOX
  - scenarioAppel :iScénario associé à l'appel.
  - fichierAppel : fichier contenant les paramètre de l'appel.

- Méthodes :
  - genererFichierAppel() : génère le fichier permettant l'appel dans le répertoire du projet (temporaire).
  - EnvoyerFichier() : permets l'envoi du fichier d'appel ou de scénario sur la trixbox via FTP.
  - run() : la classe appel héritant de QThread pour lancer le thread on utilise cette méthode.

- SuprFichierAppel() : supprime les fichiers d'appel sur la trixbox correspondant aux appels qui n'ont pas abouti
- genererUnAppel() : génère un appel.

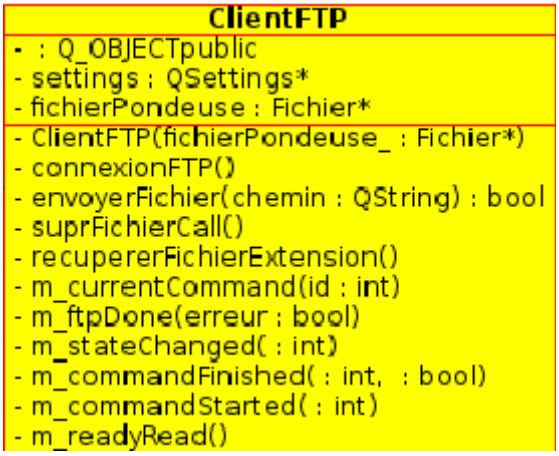
La classe Scenario :



- Attributs :
  - nom : nom du scénario
  - exist : booléen qui permet de savoir si le scénario est déjà présent dans le fichier extensions.conf qui est modifié en local dans le répertoire du projet pou l'instant.
  - numero :numero du scénario
  - fichierScenario:fichier extensions.conf qui contient les scénarios d'appel
  - ftpScenario: ftp qui fait transiter le fichier extensions.conf entre la TRIXBOX et le PC client

- Méthodes
  - genererFichierScenario():modifie le fichier extensions.conf présent en local
  - definirScenarioAppel() : récupère le fichier extensions.conf afin de le modifier à l'aide de la méthode genererFichierScénario()
- Slots :
  - m_commandFinished(int id, bool erreur) :slot exécuté a chaque fin du commande le la classe QFTP
  - m_currentCommand(int id) : affiche la commande en cours
  - m_commandStarted (int state): s 'exécute a chaque début de commande
  - m_stateChanged(int state) : affiche l'état de la connexion FTP

La Classe ClientFTP

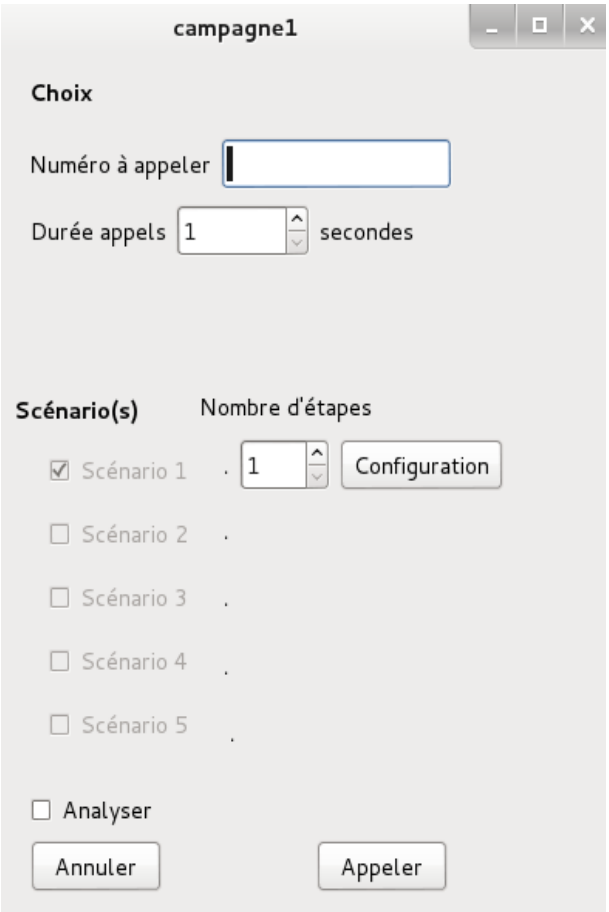


- Attributs
  - settings : instance de la classe QSettings qui permet de sauvegarder des données de connexions au FTP sur le pc.
  - fichierPondeuse : instance de la classe Fichier elle sert a faire transité les fichiers entre la TRIXBOX et le pc client.

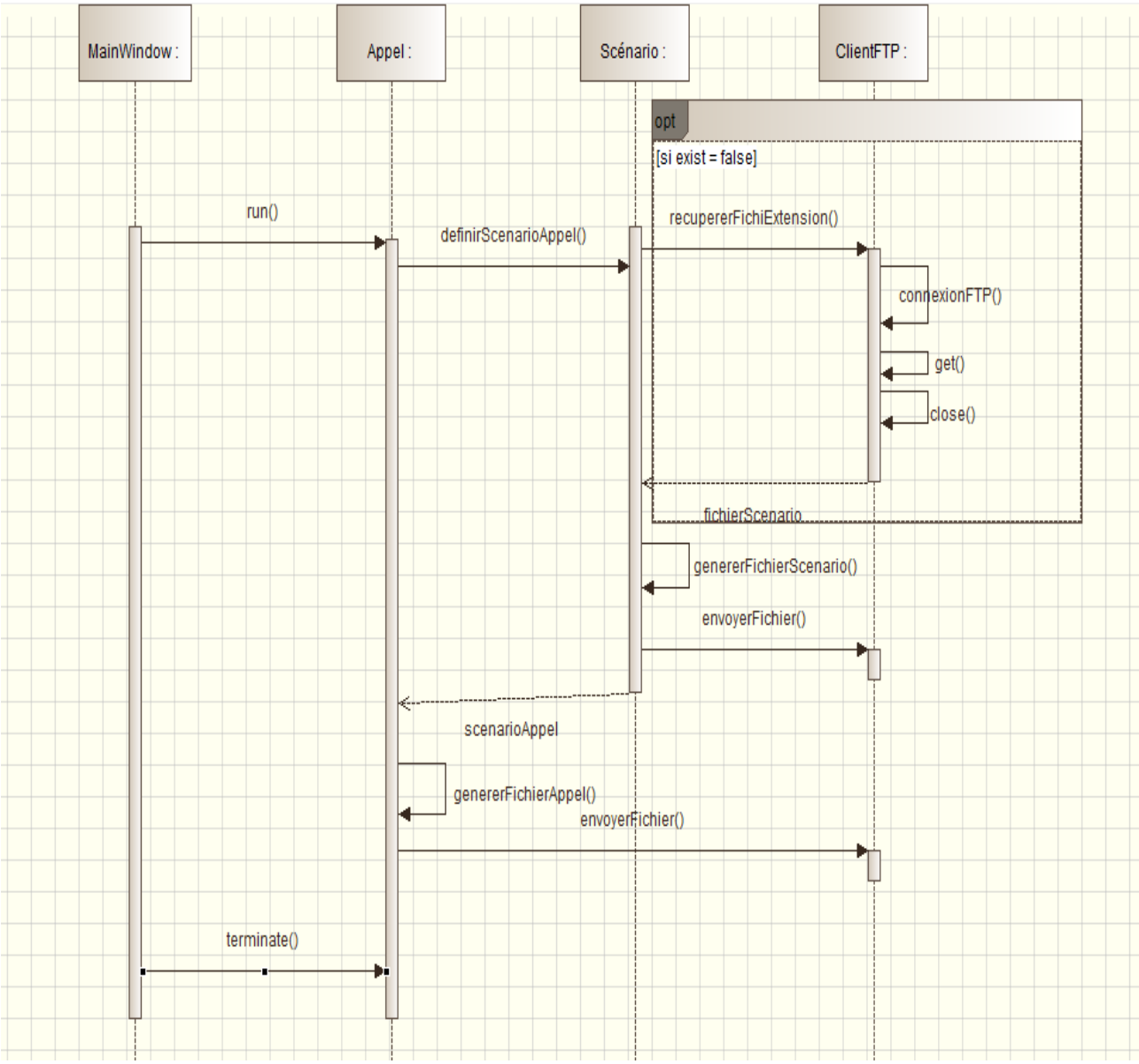
- Méthode
  - connexionFTP() : permets l'ouverture de la session FTP.
  - EnvoyerFichier() : permet l'envoi du fichier sur la trixbox (/var/spool/asterisk/outgoing/)
  - recupererFichierExtension(): télécharge le fichier extensions.conf de la trixbox sur le pc client.

8.3 Générer un appel

Après avoir configuré la campagne à l'aide de l'interface crée par l'étudiant 1(Plateau Mathias) l'utilisateur appui sur le bouton valider afin de démarrer la campagne et le cas d'utilisation générer est appel est appeler à plusieurs reprises pour générer le nombre d'appels voulu.



Le diagramme de séquence ci-dessous permet de monter comment est généré un appel sur la trixbox :



principe de fonctionnement :

lorsque le thread est lancé, on commence par définir le scénario de l'appel s'il n'existe pas.

Suite à la génération du scénario, on génère le fichier d'appel pour l'envoyer par la suite sur la trixbox pour générer l'appel

8.3.1 La génération de fichier d'appel

Pour générer un appel on utilise la fonction d'auto call de la trixbox qui grâce à un fichier rempli avec les bons paramètres génère l'appel voulu.

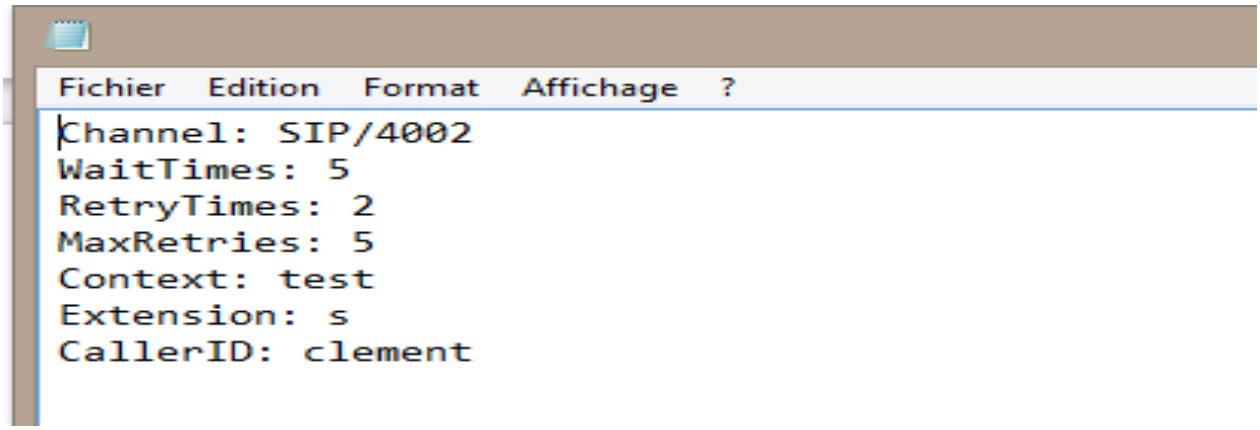
Le fichier qui doit être généré est un fichier texte brut devant devant se conformer au formalisme suivant :

- Channel : c'est le numéro qui sera appelé.
- WaitTimes:c'est le temps d'attente pour un appel sans réponse avant que l'appel

soit considéré comme « avorté ».

- RetryTimes:c'est le temps d'attente entre chaque tentative.
- MaxRetries:c'est le nombre de tentatives si la ligne n'est pas disponible.
- Context:nom du scénario qui sera défini dans le fichier extensions.conf.
- Extension:c'est là ou l'appel devra se connecter.
- CallerID:le nom qui s'affichera sur le téléphone du destinataire.

Voici un exemple de fichier call :



Pour générer le fichier de cet appel, on utilise la Classe Fichier qui hérite de la classe QFile.

Le fichier est généré dans la classe Appel qui possède un attribut de type Fichier * ce qui permet à chaque appel de générer son fichier correspondant.

Voici la méthode de la classe Appel qui génère le fichier :

```
bool Appel::genererFichierAppel()                    //génère le fichier correspondant à l'appel
{
    bool retour;
    QString nomScenario;
    nomScenario=scenarioAppel->getNom();
    if(!fichierAppel->open(QIODevice::WriteOnly | QIODevice::Text))//ouverture du fichier si il n'est pas ouvert
    {
        qDebug()<<"Impossible d'ouvrir le fichier";
        retour=false;

    }
    else
    {
        QTextStream flux(fichierAppel);              //création d'un flux pour écrire dans le fichier
        flux.setCodec("UTF-8");                      //codec utilisé
        flux <<"Channel: SIP/"<<numero<<endl<<       //ecriture dans le fichier
               "WaitTimes: "<<tempsAttenteReponse<<endl<<
               "RetryTimes: "<<tempsTentative<<endl<<
               "MaxRetries: "<<nbTentatives<<endl<<
               "Context: "<<nomScenario<<endl<<
               "Extension: s"<<endl<<
               "CallerID: "<<idAppelant<<endl;

        fichierAppel->close();                       //fermeture du fichier
        retour=true;
    }
    return retour;
}
```

Fonctionnement de la méthode :

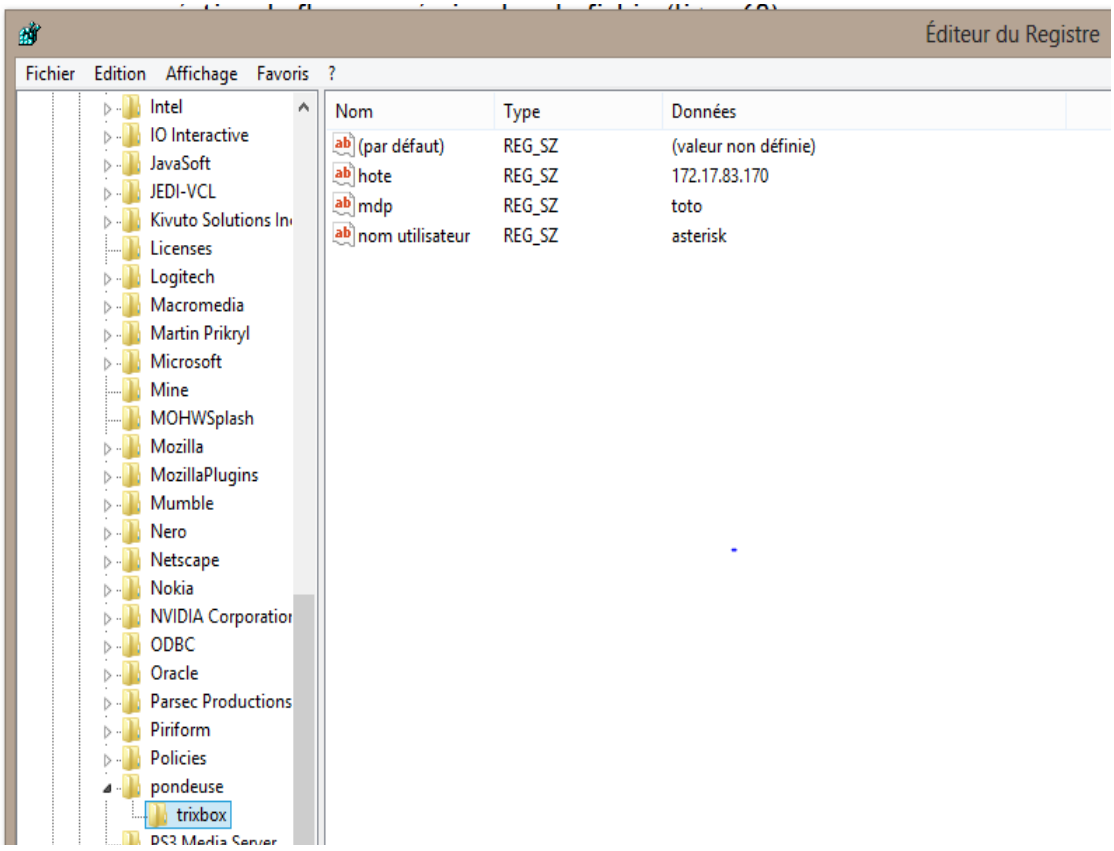
- test de l'ouverture du fichier en écriture(si le test échoue la méthode renvoie faux(ligne 61 à 64)
- création du flux pour écrire dans le fichier(ligne 69).
- Écriture des différents paramètres à l'aide des attributs de la classe Appel(ligne 71 à 77).
- Fermeture du fichier (ligne 79).

8.3.2 La connexion à la trixbox en FTP

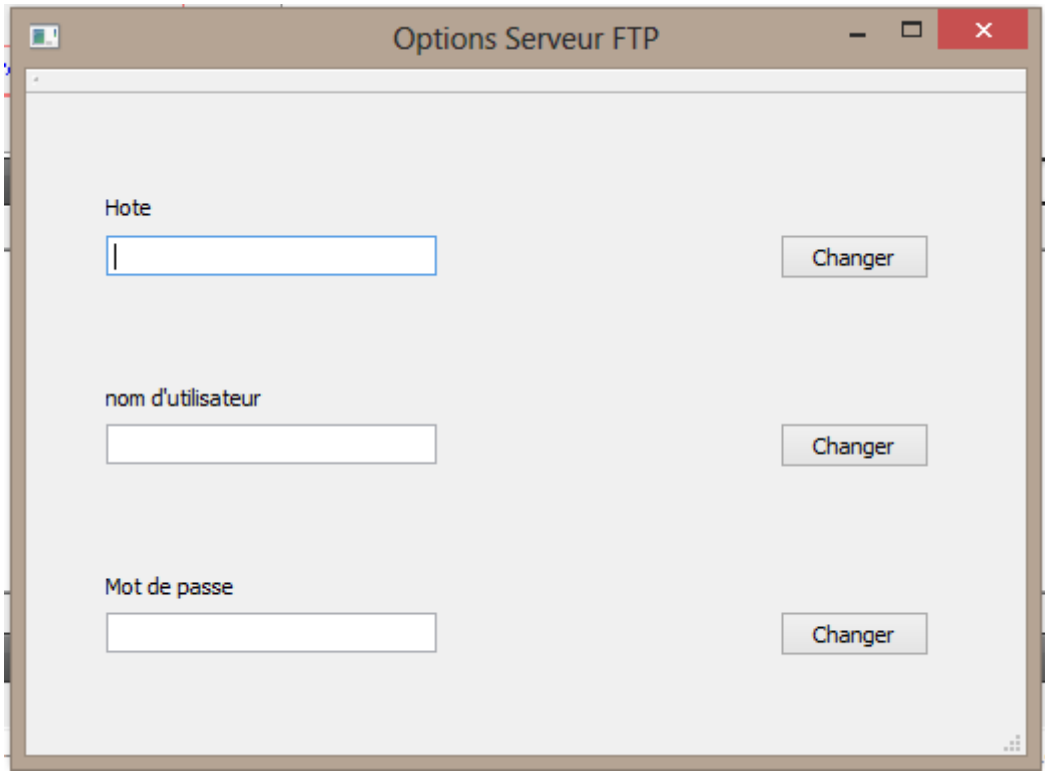
Afin de pouvoir faire transiter les fichiers entre le PC client et la TRIXBOX, l'application a besoin d'utiliser le serveur FTP(File Transfert Protocole) présent sur la TRIXBOX.

Pour se connecter au serveur, on doit connaître l'adresse IP de ce dernier, le nom d'utilisateur et le mot de passe qui correspond.

Pour faciliter la connexion et la sécuriser, une petite application a été créée pour enregistrer ces données directement dans la base de registre du système d'exploitation (pour WINDOWS).



Voici l'IHM qui permet d'enregistrer et de modifier les options qui permettent de stocker les informations nécessaires à la connexion au serveur FTP.



Lorsque l'utilisateur démarre l'application, cette fenêtre s'ouvre. Les champs Hôte, nom d'utilisateur et mot de passe sont déjà remplis avec les paramètres actuellement présent sur le PC.

Si l'utilisateur souhaite changer l'un des champs, il doit écrire les nouvelles données dans la zone de texte correspondant puis appuyer sur bouton changer en face de la ligne qui a été modifiée.

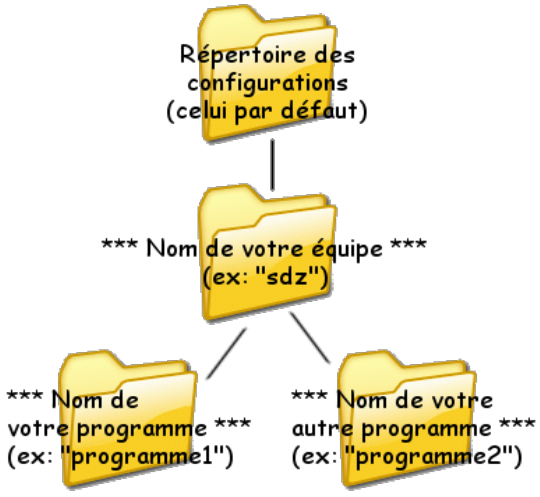
Voici le code qui permet de modifier les paramètres de connexion au serveur Trixbox

```
4   MainWindow::MainWindow(QWidget *parent) :
5       QMainWindow(parent),
6       ui(new Ui::MainWindow)
7   {
8       ui->setupUi(this);
9       settings=new QSettings("pondeuse", "trixbox");
10      ui->lineEditUsers->setText(settings->value("nom utilisateur").toString());
11      ui->lineEditHote->setText(settings->value("hote").toString());
12      ui->lineEditMDP->setText(settings->value("mdp").toString());
13
14  }
15
16  MainWindow::~MainWindow()
17  {
18      delete ui;
19  }
20
21  void MainWindow::on_pushButtonHote_clicked()
22  {
23      settings->setValue("hote", ui->lineEditHote->text());
24      ui->lineEditHote->clear();
25  }
26
27  void MainWindow::on_pushButtonUsers_clicked()
28  {
29      settings->setValue("nom utilisateur", ui->lineEditUsers->text());
30      ui->lineEditUsers->clear();
31  }
32
33  void MainWindow::on_pushButton_clicked()
34  {
35      settings->setValue("mdp", ui->lineEditMDP->text());
36      ui->lineEditMDP->clear();
37  }
```

Ligne 9 :on instancie un objet QSettings qui créer un champ dans le registre.

1Er argument : nom de l'équipe

2e argument : non du programme


Répertoire des configurations (celui par défaut)
*** Nom de votre équipe *** (ex: "sdz")
*** Nom de votre programme *** (ex: "programme1")
*** Nom de votre autre programme *** (ex: "programme2")

Cette ligne permet dans un premier cas si les entrées n'existent pas ce la les créent et s'il existe cela récupère les données déjà présentent.

Lignes 10 à 12 :ces lignes permettent de mettre les valeurs qui sont présentent actuellement dans le registre dans les zones de texte correspondantes.

Lignes 21 à 37:ces slots permettent de changer les valeurs dans le registre à l'aide de la fonction setValue() ;

Après l'enregistrement on efface la LineEdit avec la fonction clear() ;

8.3.3 Application dans le programme principal de la pondeuse d'appels

Voici la méthode de la classe ClientFTP qui permet la connexion au serveur FTP de la trixbox.

```
150 void ClientFTP::connexionFTP()
151 {
152     this->connectToHost(settings->value("hote").toString());
153     this->login(settings->value("nom utilisateur").toString(),settings->value("mdp").toString());
154 }
155
```

Les lignes settings->value(« ... ») correspondent aux champs dans le registre qui contiennent les informations.

Ce sont des méthodes qui font partie de la classe Qsettings qui permettent de récupérer les valeurs présentes dans la base du registre.

8.3.4 La génération d'un scénario

Le scénario d'un appel correspond à la suite d'actions qui seront effectuées pendant l'appel et qui sont défini dans l'IHM de Mr Mathias Plateau:

- décrocher
- pause de 3 secondes
- envoi du DTMF #
- pause de 1 seconde
- envoi du DTMF 3 et 4
- pauses de 5 secondes
- lecture du fichier goodbye.ulaw
- raccrocher

L'appel se fera via un fichier call présent sur la TRIXBOX.

Pour rappel voici la structure de ce fichier.

- Channel : c'est le numéro qui sera appelé.
- WaitTimes:c'est le temps d'attente pour un appel sans réponse avant que l'appel soit considéré comme « avorté ».
- RetryTimes:c'est le temps d'attente entre chaque tentative.
- MaxRetries:c'est le nombre de tentatives si la ligne n'est pas disponible.
- Context:nom du scénario qui sera défini dans le fichier extensions.conf.
- Extension:c'est là ou l'appel devra se connecter.
- CallerID:le nom qui s'affichera sur le téléphone du destinataire.

Comme chaque appel est différent, c'est la ligne « context » qui va changer. Chaque « context » correspond à un scénario et tous les « contexts » sont mis dans le fichier extensions.conf du répertoire, /etc/asterisk.

Pour générer un scénario, il faut d'abord télécharger le fichier extensions.conf qui se situe sur la trixbox pour le placer sur le pc client afin de pouvoir le modifier en local.

Le fichier extensions.conf est structuré de la manière suivante :

```
[general]

[globals]

[nom d'un scénario]

exten => X, Y, Z
```

general : ce sont les option du fichier.

globals : vous pouvez spécifier vos propres variables, qui peut être utilisé plus tard dans les extensions.

Ce qui n'est ni « general », ni « globals » est considéré comme un scénario.

X : extensions numéro pour lequel le scénario s'applique, s signifie toute les extensions

Y : priorité de l'action

Z : fonction qui sera effectué

8.3.5 recupererFichierExtension()

Le téléchargement du fichier extensions.conf qui se trouve dans le répertoire /,etc/asterisk se fait via le méthode XXX de la classe YYY.

Voici le code de la méthode :

```
187 void ClientFTP::recupererFichierExtension()        //télécharge le fichier extensions.conf de la trixbox
188 {
189
190     this->connexionFTP();
191     this->cd("/etc/asterisk/");
192     this->get("extensions.conf",fichierPondeuse);
193     qDebug()<<"fichier telecharge";
194     this->close();
195 }
```

Fonctionnement de la méthode :

- ligne 190 : connexion au serveur ftp de la trixbox.
- Ligne 192 : Changement de répertoire pour se placer dans le répertoire du fichier extensions.conf (/etc/asterisk/).
- Ligne 193 : Récupération du fichier extensions.conf et mise à jour de l'attribut fichierPondeuse.
- Ligne 195 : fermeture de la connexion FTP.

Suite à la récupération du fichier extensions.conf, nous allons le modifier dans le but d'inscrire un context dans le fichier qui correspondra au scénario que l'appel devra effectuer.

8.3.6 genererFichierScenario() :

Cette méthode fait partie de la classe Scenario elle a pour but de modifier le fichier extensions.conf . Elle modifie ce fichier afin de créer un scénario qui correspond à ce que souhaite l'utilisateur.

```
185 void Scenario::genererFichierScenario()
186 {
187     QVector<etape>::iterator i=tableauEtapes->begin();
188     if(!fichierScenario->open(QIODevice::Append | QIODevice::Text))
189     {
190         qDebug()<<"impossible d'ouvrir le fichier"<<endl;
191         return;
192     }
193     else
194     {
195         qDebug()<<"generation fichier scenario";
196         QTextStream flux(fichierScenario);
197         flux.setCodec("UTF-8");
198         flux<<"["<<nom<<"]"<<endl
199            <<"exten=> 1,n, Answer()"<<endl;
200
201         do
202         {
203             qDebug()<<"ecriture";
204             flux << "exten => s,n,SendDTMF("<< (*i).valeur <<")"<<endl
205                  <<"exten => s,n,wait("<< (*i).tempsPause <<")"<<endl;
206             i++;
207         }while (i != tableauEtapes->end());
208         flux<<"exten => s,n,Hangup()"<<endl<<endl;
209         fichierScenario->close();
210     }
211 }
```

Fonctionnement de la méthode genererFichierScenario() :

- test de l'ouverture de fichier en mode ajout afin de ne pas supprimer les autres données du fichier
- ouverture d'un flux d'écriture
- choix du codec UTF-8
- écriture d'un context :

  la ligne « exten=> s,n Answer() » sert à décrocher

  L'utilisateur a plusieurs choix :

  - envoyer un DTMF(Dual-tone multi-frequency signaling)
  - « exten=> s,n,SendDTMF(valeur du ou des DTMFs) »
  - faire une pause

    « exten=> s,n,Wait(temps de pause en seconde) »
  - raccrocher

    « exten=> s,n,Hangup() »

Le « s » signifie que l'action s'effectue sur n'importe quelle numérotation

le « n » indique que c'est la prochaine étape à effectué.On peut placer un chiffre à la place du « n » qui permettra de donner une priorité à l'action associée.

Après avoir généré le fichier d'appel (fichier call), modifier le fichier extensions.conf, il faut uploader ces deux fichiers dans les répertoires appropriés.

/etc/asterisk pour le fichier extensions.conf et

/var/spool/asterisk/outgoing pour le fichier d'appel (.call)

l'upload doit se faire dans un ordre précis le fichier scénario doit être uploader avant le fichier d'appel, car dans le cas contraire l'appel ne pourra pas exécuté le scénario s'il n'existe pas.

Afin d'uploader ces deux fichiers, on utilisera la méthode **envoyerFichier(QString chemin)**.

La méthode prend en paramètre un QString qui est le chemin du répertoire ou le fichier sera envoyé. Les deux chemins possibles sont mis dans des constantes.

Voici la méthode :

```
156 bool ClientFTP::envoyerFichier(QString chemin)//place un fichier sur le serveur en FTP
157 {
158     bool retourFichierCall;
159     retourFichierCall=false;
160     if (fichierPondeuse->open(QIODevice::ReadOnly))
161     {
162         this->connexionFTP();
163         this->put(fichierPondeuse,chemin+fichierPondeuse->fileName());
164         retourFichierCall=true;
165     }
166
167     if(retourFichierCall==false)
168     {
169         qDebug()<<"Fichier non envoyer"<<endl;
170     }
171     else
172     {
173         qDebug()<<"Fichier envoyer"<<endl;
174     }
175     this->close();
176
177     return retourFichierCall;
178 }
179
```

Fonctionnement de la méthode :

- on commence par tester l'ouverture du fichier en mode lecture. (ligne 160 à 165)
- On effectue une connexion en FTP sur le serveur à l'aide de la méthode connexionFTP()(ligne 162)
- ensuite on transfert le fichier sur le serveur à l'aide de la méthode **put(&File, QString) ;**(ligne 163)
- fin de la connexion (l96)(ligne 175)

Suite au téléchargement du fichier d'appel (.call) la trixbox génère l'appel au destinataire voulu et avec le scénario défini dans le fichier extensions.conf.

Si le fichier .call possède les bons paramètres, l'appel est effectué et le fichier est automatiquement supprimé par le système (asterisk).

Si l'appel ne possède pas les paramètres requis, le fichier est supprimé sans effectuer l'appel.

Si la ligne est occupée, le fichier reste dans le répertoire .

8.4 Conclusion :

La partie du projet que j'avais en charge consistait à générer des appels en simultané sur un serveur d'appels prêté pour cette occasion par les MMA . Chaque appel devait suivre un scénario bien défini par l'utilisateur.

Au moment de la rédaction de ce dossier, la majeure partie des fonctionnalités qui m'incombaient sont opérationnelles.

La génération d'un appel avec une liste d'action (DTMF)fonctionne.

La génération d'appel simultané fonctionne (appel sur plusieurs postes).

En revanche des parties du projet n'ont pas encore été traitées, notamment à récupération des appels qui n'ont pas abouti.

La mise en commun de la partie qui génère l'appel et l'IHM qui configure une campagne n'est pas encore faites.

Par conséquent, je n'ai pas réussi à effectuer une campagne complète.

Le test de la surcharge du serveur n'est donc pas encore effectif.

Ce retard s'explique par la difficulté que nous avons eue au niveau de la compréhension et de la mise en place des appels automatiques.

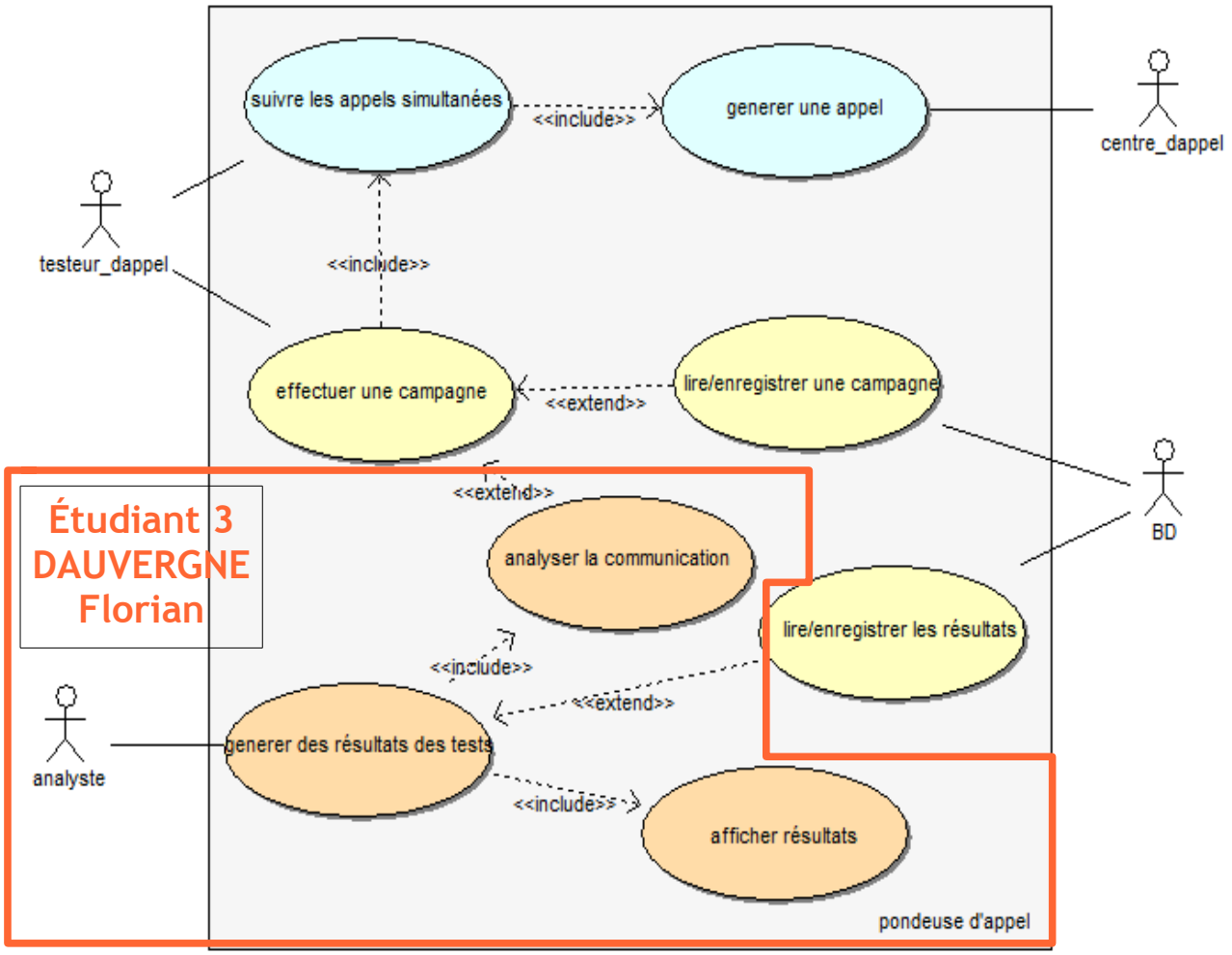
Le transfert des fichiers entre la trixbox et le pc de l'utilisateur grâce au protocole FTP a été très long à mettre en place (compréhension et appropriation de la bibliothèque QT associée) et a donc retardé la suite du projet.

Dans l'ensemble ce projet a été bénéfique le fait d'avoir une tâche bien définie m'a beaucoup apporté au niveau de l'autonomie.

Le fait de travaillé en équipe a aussi été bénéfique malgré le fait que je n'ai pas travaillé avec les personnes que je souhaitais le projet s'est passé dans de bonnes conditions, il y a du dialogue dans l'équipe chaque personne du groupe sait ce qu'il a à faire et est motivé pour le faire.

Ce Projet m'a permis de me conforter dans le choix de ma poursuite d'étude dans le développement et il m'a permis d'approfondir mes connaissances dans le développement en C++ QT.

9. Conception détaillée : Partie Dauvergne Florian (E3)

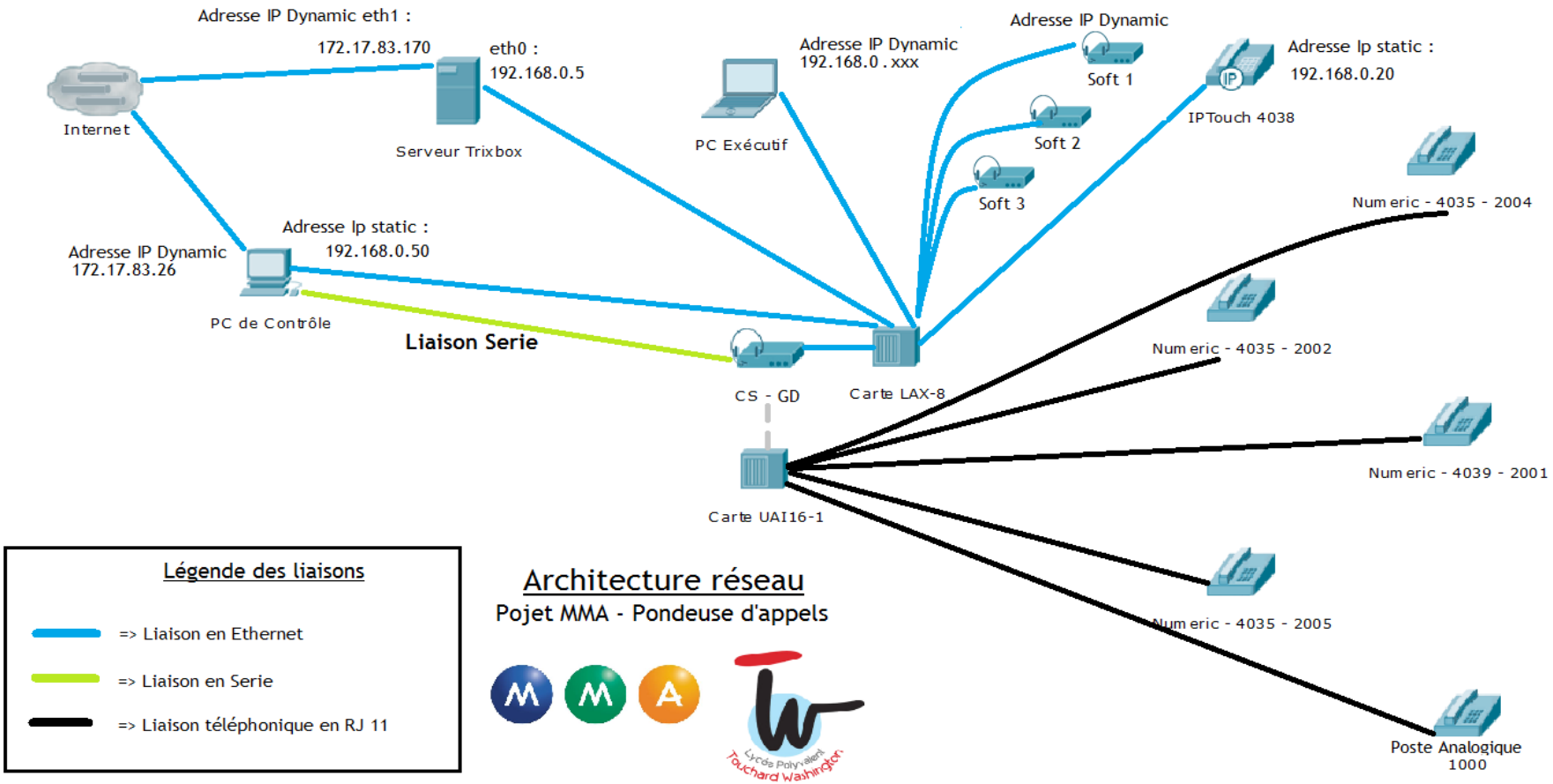


9.1 Configuration analyse

Nous devons analyser la communication d'une campagne d'appels. Pour cela nous devons capturer des paquets afin de récupérer les paramètres permettant l'analyse de cette campagne.

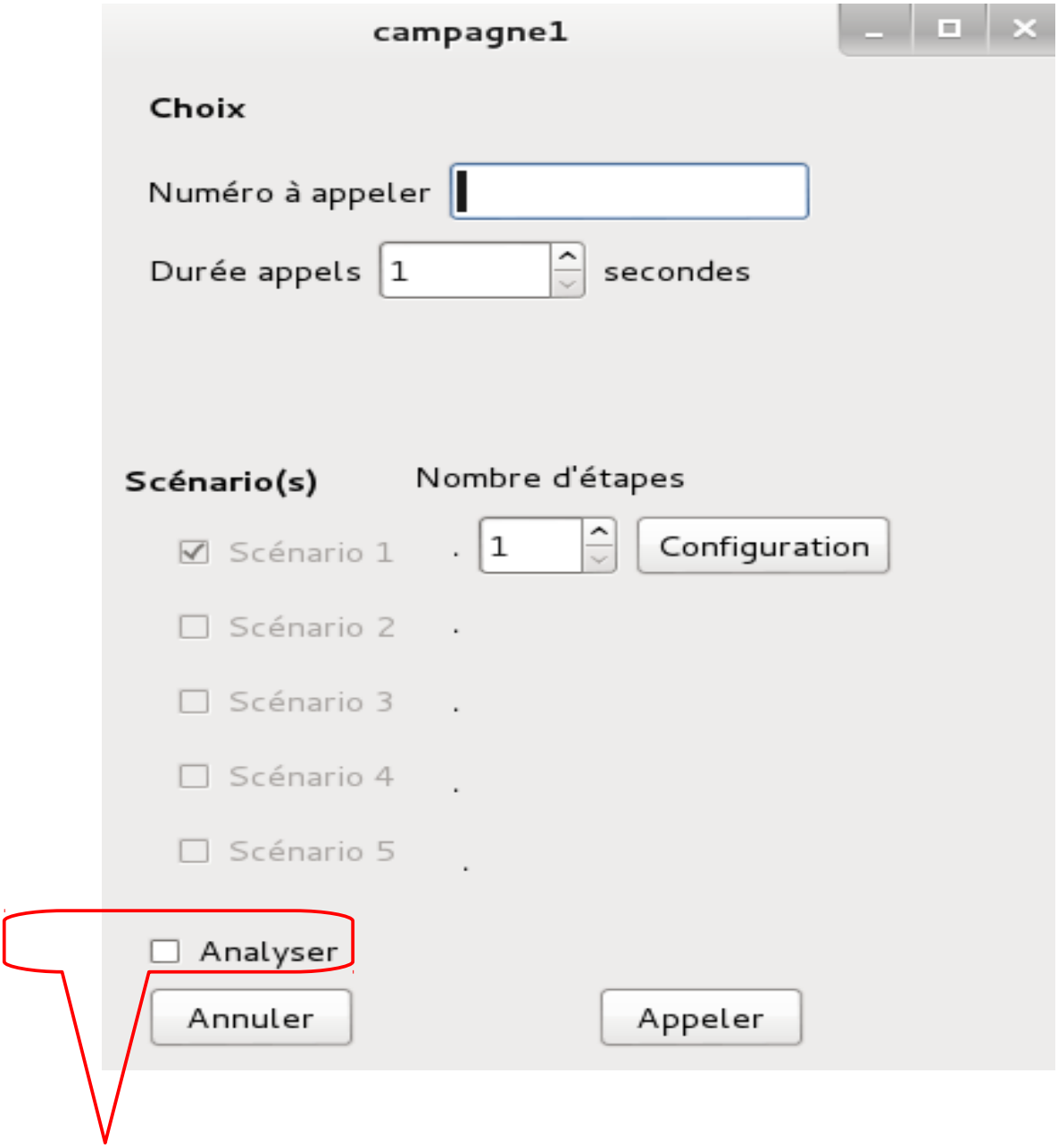
9.2 Pré-Requis

L'application doit se trouver sur un hub situé entre l'appelant et l'appelé car il va nous permettre de capturer tout le trafic et de filtrer seulement ce qui nous intéressent.



9.3 Lancement analyse

Le lancement de l'analyse se fera à partir de l'IHM de Mathias Plateau. Lors de la configuration d'une campagne d'appel, l'utilisateur pourra choisir ou non d'analyser celle-ci.



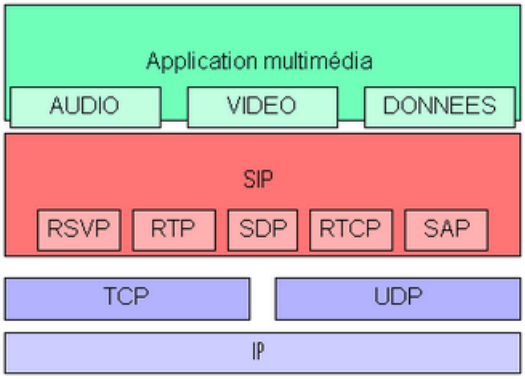
L'utilisateur aura à cocher cette cache, lorsqu'il la cochera, l'analyse débutera en même temps que la campagne d'appels.

9.4 Analyse de la qualité de service

9.4.1 Fonctionnement et mise en œuvre

Le transfert de paquet est le moyen de transmission d'un message sur un réseau. Le message n'est jamais envoyé en un seul bloc, toutes les informations à transporter sont découpées en paquets pour être acheminés d'un bout à l'autre du réseau. Le paquet de base consiste en un en-tête avec les adresses des systèmes émetteur et récepteur, ainsi qu'un corps, ou champ de données, avec les données à transférer. Lorsque le paquet parcourt la pile de protocoles TCP/IP, les protocoles de chaque couche ajoutent ou suppriment des champs de l'en-tête de base. Lorsqu'un protocole sur le système émetteur ajoute des données à l'en-tête du paquet, le processus s'appelle encapsulation de données.

Voici un petit schéma explicatif d'encapsulation des paquets :



Encapsulation des paquets

Ainsi, afin d'analyser la communication, nous allons devoir capturer et décoder les paquets transitant sur le réseau.

On utilisera pour cela la librairie pcap, étant déjà utilisé par l'analyseur réseau wireshark, elle répondra donc à nos besoins.

Il nous faudra donc déterminer quels paquets contiennent les informations de gigue, perte de paquets, d'écho, et de délai d'acheminement. D'après le schéma d'encapsulation, on s'est d'abord intéressé aux paquets IP puis nous sommes remontés jusqu'en de la pile.

Commençons par le paquet IP :

32 Bits			
Version	Longueur d'en tête	Type de service	Longueur totale
Identification		Drapeau	Décalage fragment
Durée de vie	Protocole		Somme de contrôle d'en tête
Adresse IP source			
Adresse IP destination			
Données			

Signification des différents champs :

- **Version**(4 bits) : il s'agit de la version du protocole IP que l'on utilise (actuellement on utilise la version 4 *IPv4*) afin de vérifier la validité du datagramme. Elle est codée sur 4 bits.
- **Longueur d'en-tête**, ou *IHL* pour *Internet Header Length* (4 bits) : il s'agit du nombre de mots de 32 bits constituant l'en-tête (nota : la valeur minimale est 5). Ce champ est codé sur 4 bits.
- **Type de service**(8 bits) : il indique la façon selon laquelle le datagramme doit être traité.
- **Longueur totale** (16 bits): il indique la taille totale du datagramme en octets. La taille de ce champ étant de 2 octets, la taille totale du datagramme ne peut dépasser 65536 octets. Utilisé conjointement avec la taille de l'en-tête, ce champ permet de déterminer où sont situées les données.
- **Identification, drapeaux (flags) et déplacement de fragment** sont des champs qui permettent la fragmentation des datagrammes, ils sont expliqués plus bas.
- **Durée de vie** (8 bits) : ce champ indique le nombre maximal de routeurs à travers lesquels le datagramme peut passer. Ainsi ce champ est décrémenté à chaque passage dans un routeur, lorsque celui-ci atteint la valeur critique de 0, le routeur détruit le datagramme. Cela évite l'encombrement du réseau par les datagrammes perdus.
- **Protocole** (8 bits) : ce champ, en notation décimale, permet de savoir de quel protocole est issu le datagramme
  - ICMP : 1
  - IGMP : 2
  - TCP : 6
  - UDP : 17
- **Somme de contrôle de l'en-tête, ou en anglais *header checksum* (16 bits)** : ce champ contient une valeur codée sur 16 bits qui permet de contrôler l'intégrité de l'en-tête afin de déterminer si celui-ci n'a pas été altéré pendant la transmission. La somme de contrôle est le complément à un de tous les mots de 16 bits de l'en-tête (champ *somme de contrôle* exclu). Celle-ci est en fait telle que lorsque l'on fait la somme des champs de l'en-tête (somme de contrôle incluse), on obtient un nombre avec tous les bits positionnés à 1
- **Adresse IP source** (32 bits) : Ce champ représente l'adresse IP de la machine émettrice, il permet au destinataire de répondre
- **Adresse IP destination** (32 bits) : adresse IP du destinataire du message

Nous savons que les paquets RTP et RTCP (ceux dont nous avons besoin) sont encapsulés dans UDP. Regardons donc un paquet UDP.

En tête paquet UDP

L'en-tête du segment UDP est donc très simple :

Port Source(16 bits)	Port Destination(16 bits)
Longueur(16 bits)	Somme de contrôle(16 bits)
Données(longueur variable)	

Le ***User Datagram Protocol*** (UDP, en français **protocole de datagramme utilisateur**) est un des principaux protocoles de télécommunication utilisés par Internet. Le rôle de ce protocole est de permettre la transmission de données de manière très simple entre deux entités, chacune étant définie par une adresse IP et un numéro de port.

Signification des différents champs :

- **Port Source**: il s'agit du numéro de port correspondant à l'application émettrice du segment UDP. Ce champ représente une adresse de réponse pour le destinataire. Ainsi, ce champ est optionnel, cela signifie que si l'on ne précise pas le port source, les 16 bits de ce champ seront mis à zéro, auquel cas le destinataire ne pourra pas répondre (cela n'est pas forcément nécessaire, notamment pour des messages unidirectionnels.

- **Port Destination**: Ce champ contient le port correspondant à l'application de la machine destinataire à laquelle on s'adresse.

- **Longueur**: Ce champ précise la longueur totale du segment, en-tête comprise, or l'en-tête a une longueur de 4 x 16 bits (soient 8 x 8 bits) donc le champ longueur est nécessairement supérieur ou égal à 8 octets.

- **Somme de contrôle**: Il s'agit d'une somme de contrôle réalisée de telle façon à pouvoir contrôler l'intégrité du segment.

Plus concrètement voilà ce que contient un paquet capturés par Wireshark fonctionnant avec pcap :

```
▽ User Datagram Protocol, Src Port: 58453 (58453), Dst Port: 32137 (32137)
    Source port: 58453 (58453)
    Destination port: 32137 (32137)
    Length: 38
  ▷ Checksum: 0xc8f6 [validation disabled]
```

Il nous faut maintenant déterminer les paquets RTP et RTCP encapsulés dans UDP. RTP et RTCP utilisent des ports différents. RTP utilise un numéro de port pair, et RTCP le

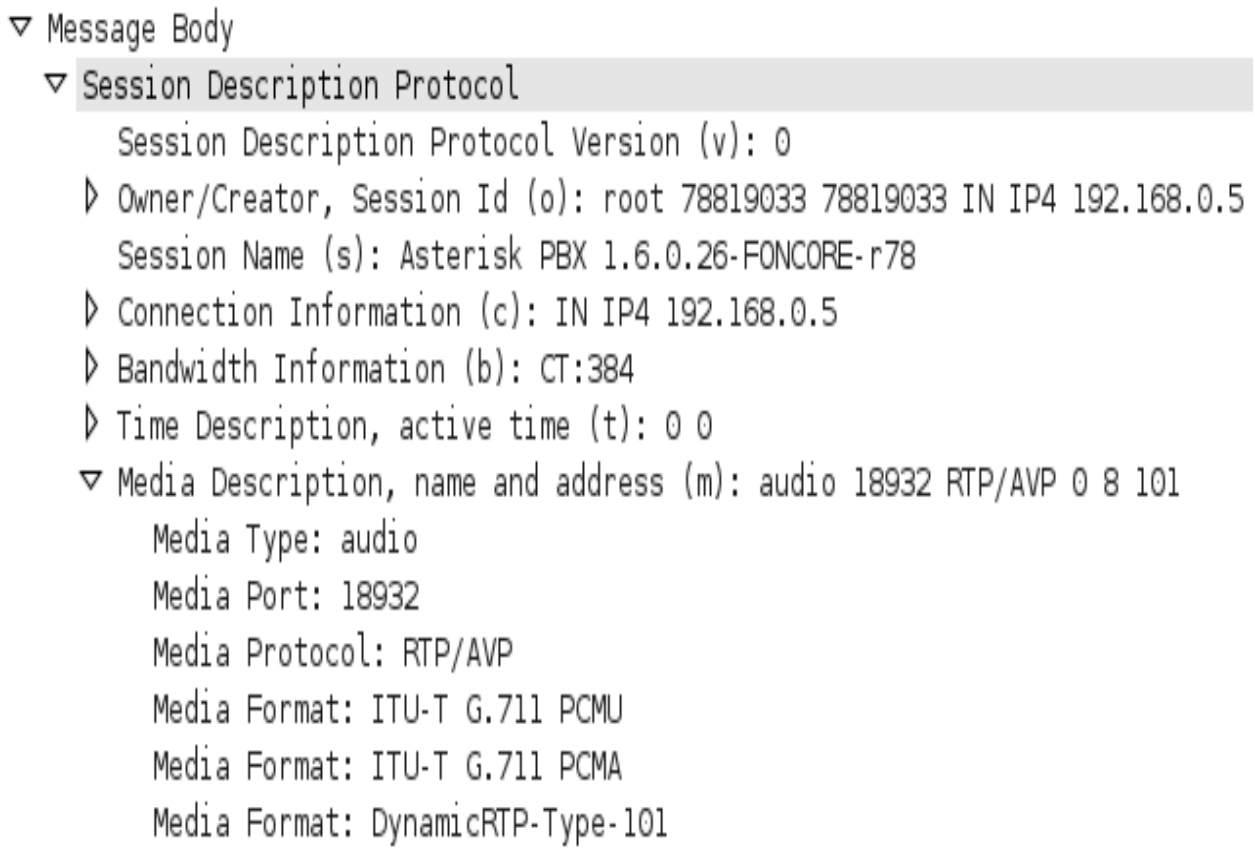
numéro de port impair qui suit directement. Les numéros de port utilisés par RTP et RTCP sont compris entre 1025 et 65535. Les ports RTP et RTCP par défaut sont respectivement 5004 et 5005. Afin de pouvoir filtrer les paquets RTP il nous faut donc déterminer quels ports sont utilisés sur la campagne.

On a donc chercher à savoir quel paquet encapsulés dans UDP contenait cette information. Après analyse des paquets SIP/SDP on s'aperçoit qu'ils contiennent cette information.

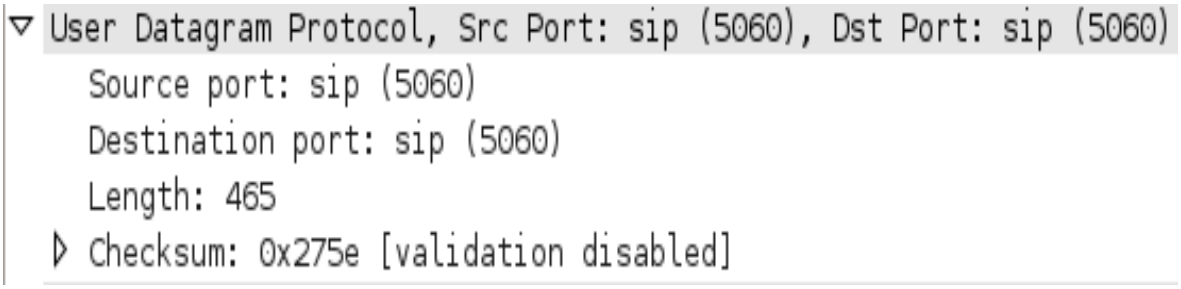
Session Initiation Protocol (SIP) est un protocole standard ouvert de gestion de sessions souvent utilisé dans les télécommunications multimédia (son, image, etc.). Il se charge de l'authentification et de la localisation des multiples participants. Il se charge également de la négociation sur les types de média utilisables par les différents participants en encapsulant des messages SDP (Session description protocol). SIP utilise le port 5060, port attribué par l'organisme IANA.

Dans le cas de paquets SIP caractéristiques d'un appel, nous extrayons les données SDP et en déduisons les numéros de ports qui seront utilisés pour le trafic RTP/RTCP.

Le logiciel sait alors quels paquets il doit intercepter pour avoir accès aux données de la conversation.


Message Body
Session Description Protocol
Session Description Protocol Version (v): 0
Owner/Creator, Session Id (o): root 78819033 78819033 IN IP4 192.168.0.5
Session Name (s): Asterisk PBX 1.6.0.26-FONCORE-r78
Connection Information (c): IN IP4 192.168.0.5
Bandwidth Information (b): CT:384
Time Description, active time (t): 0 0
Media Description, name and address (m): audio 18932 RTP/AVP 0 8 101
Media Type: audio
Media Port: 18932
Media Protocol: RTP/AVP
Media Format: ITU-T G.711 PCMU
Media Format: ITU-T G.711 PCMA
Media Format: DynamicRTP-Type-101

Comme le montre l'encadré, c'est dans le paquet SIP/SDP que l'on détermine sur quel port transit le paquet RTP, ici 18932. Le paquet RTCP sera donc port 18933. Néanmoins on ne sait pas filtrer directement les paquets SIP mais on sait qu'ils utilisent le port 5060, on va donc filtrer à partir des paquets UDP les paquets sur le port 5060.


User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Source port: sip (5060)
Destination port: sip (5060)
Length: 465
Checksum: 0x275e [validation disabled]

Comme le montre cette capture, on détermine le paquet SIP à partir d'un paquet UDP et du port 5060.

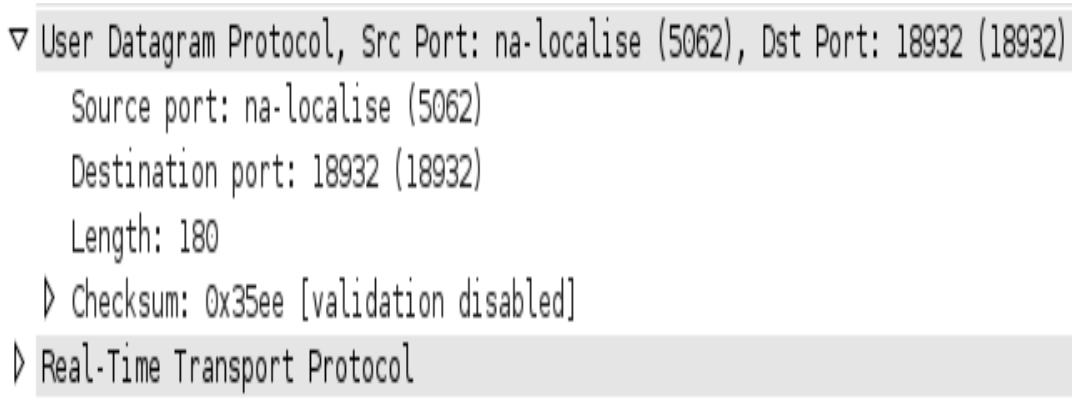
Récapitulatif :

Nous devons filtrer les paquets RTP et RTCP. Or nous ne sommes pas en mesure d'appliquer ce filtre d'entrée.

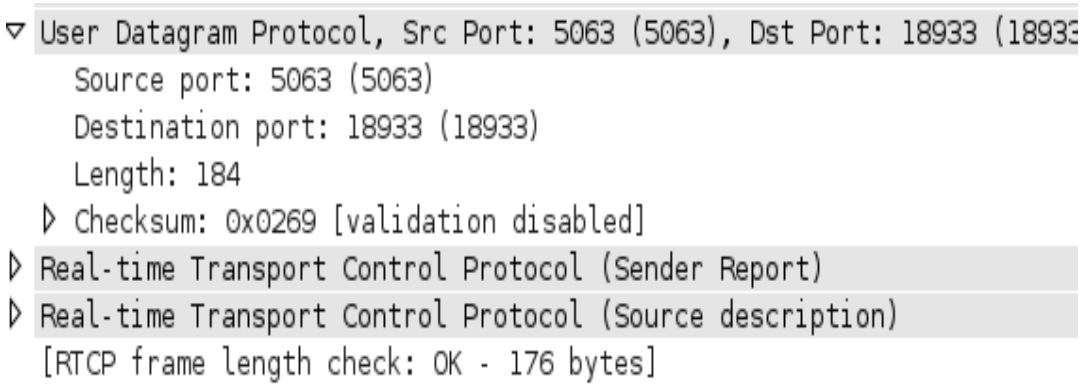
Pour cela on doit filtrer les paquets SIP et récupérer le numéro de port RTP. Nous ne pouvons pas appliquer un filtre SIP d'entrée on filtre donc les paquets UDP avec le port 5060.

Nous obtenons des paquets SIP que nous décryptons pour obtenir le numéro de port RTP.

Maintenant que l'on connaît le numéro de port récupéré dans les paquets SIP, on va une nouvelle fois appliquer un filtre UDP, les paquets RTP et RTCP étant encapsulés dans UDP, on filtre afin de n'avoir que les paquets dont nous avons besoin pour identifier les paquets RTP.


User Datagram Protocol, Src Port: na-localise (5062), Dst Port: 18932 (18932)
Source port: na-localise (5062)
Destination port: 18932 (18932)
Length: 180
Checksum: 0x35ee [validation disabled]
Real-Time Transport Protocol

Paquet RTP


User Datagram Protocol, Src Port: 5063 (5063), Dst Port: 18933 (18933
Source port: 5063 (5063)
Destination port: 18933 (18933)
Length: 184
Checksum: 0x0269 [validation disabled]
Real-time Transport Control Protocol (Sender Report)
Real-time Transport Control Protocol (Source description)
[RTCP frame length check: OK - 176 bytes]

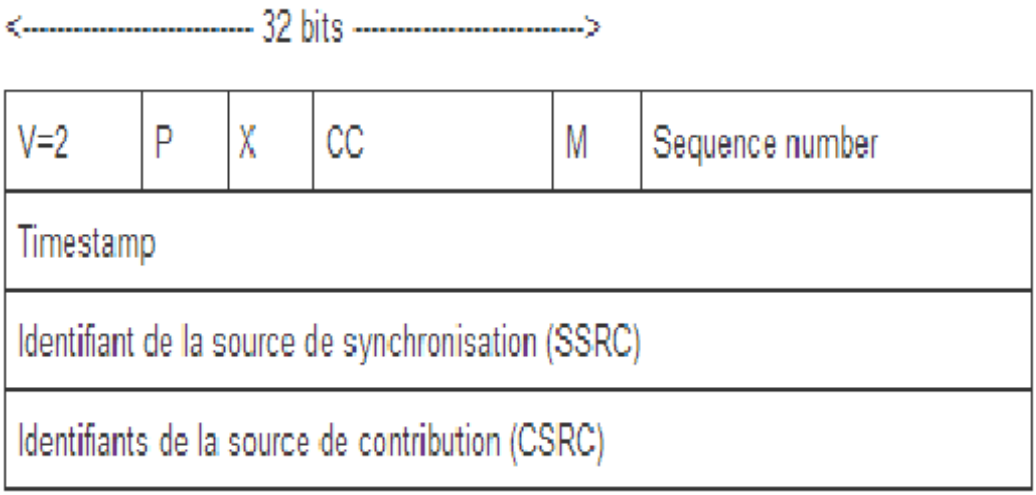
Paquet RTCP

Nous capturons les paquets UDP, comme le montre cette capture de wireshark, les champs de port source et destination nous fournissent les informations nécessaires à l'identification des paquets RTP et RTCP.

Nous avions récupéré dans les paquets SIP le port 18932, ce qui correspond ici. Ce paquet est donc identifié comme un paquet RTP et l'autre RTCP.

Comme nous avons vu plus haut le numéro de port récupéré est 18932 et 5062 pour les paquets RTP. Comme on le voir ici les paquets sont bien identifié et les numéros de port correspondent. On peut donc filtrer les paquets RTP avec ces informations.

En tête d'un paquet RTP



Maintenant nous savons filtrer les paquets RTP et RTCP qui contiennent les informations nécessaires à l'analyse de la communication. Nous connaissons les ports où transitent les paquets, on peut donc récupérer les paquets RTP grâce au numéro de port, et les RTCP grâce au numéro de port +1.

Le protocole RTP est un protocole temps réel non fiable qui permet de transporter des informations multimédia. Le protocole RTP fournit :

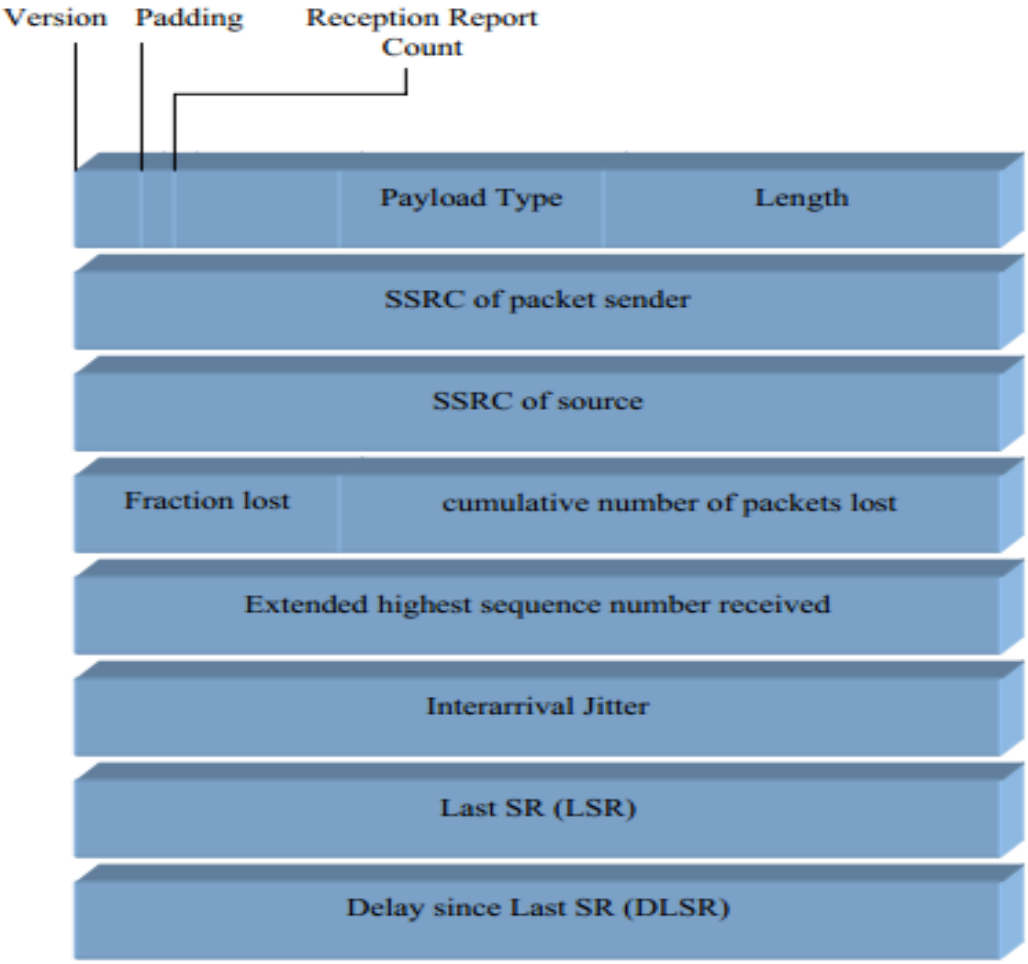
- L'identification du type des informations multimédia,
- La numérotation du séquencement,
- L'horodatage des paquets RTP émis.

RTCP est le protocole de contrôle associé au protocole RTP. Les paquets RTCP sont émis périodiquement et véhiculent des informations statistiques sur l'état de la transmission entre chaque participant.

Ces informations statistiques permettent de calculer des informations de QoS :

- Gigue : temps d'inter arrivée, permet d'estimer l'écart entre l'arrivée des paquets et leur moment de lecture,
- RTT : durée que met un paquet pour faire un aller retour sur le réseau,
- Taux de perte de paquet : estimation du taux de paquet perdu.

En tête d'un paquet RTCP



Le champ cumulative number of packets lost (24 bits) qui nous permet de savoir le nombre de paquets perdus.

Le champ interarrival jitter (32 bits). C'est une estimation de l'intervalle de temps d'un packet de donnés RTP qui est mesuré avec le timestamp et qui est sous forme d'un entier. C'est en fait le temps relatif de transit entre deux paquets de donnés.

Pour effectuer l'analyse de la communication, nous disposons de deux classes. La première une classe capture, qui contient nos fonctions de capture et nos fonctions de filtrage. Puis une deuxième classe qui va nous permettre d'afficher les statistiques de la communication.

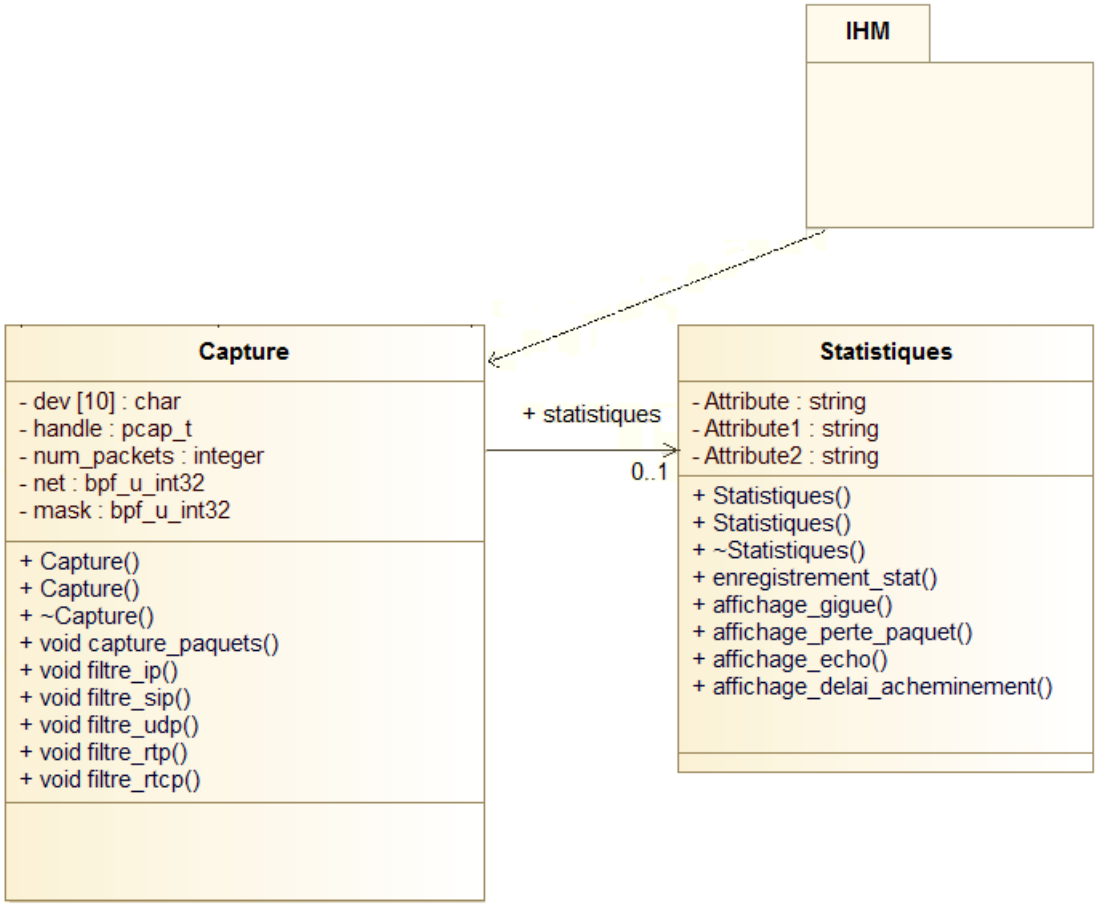


Diagramme de classe

9.4.2 Capture

La capture se fait en parallèle de la campagne. Elle va nous permettre de récupérer les informations nécessaires à l'analyse de la QoS d'une campagne d'appels. La capture de paquets des paquets se fait à l'aide de la bibliothèque pcap.

9.4.3 Capture

La capture se fait en parallèle de la campagne. Elle va nous permettre de récupérer les informations nécessaires à l'analyse de la QoS d'une campagne d'appels.

La capture de paquets des paquets se fait à l'aide de la bibliothèque pcap et de la fonction capture_paquets

```
26  void Capture::capture_paquets()
27  {
28
29      char errbuf[PCAP_ERRBUF_SIZE];  /* erreur buffer */
30
31      /* print capture info */
32      printf("Périphérique: %s\n", dev);
33      printf("Nombre de paquets: %d\n", num_packets);
34      printf("Filter expression: %s\n", filter_exp);
35
36      /* ouvre un périphérique de capture */
37      handle = pcap_open_live(dev, SNAP_LEN, 1, 1000, errbuf);
38
39      if (handle == NULL)
40      {
41          fprintf(stderr, "Impossible d'ouvrir le périphérique' %s: %s\n", dev, errbuf);
42          exit(EXIT_FAILURE);
43      }
44
45      /* Assurez-vous que nous capturons sur un périphérique Ethernet */
46      if (pcap_datalink(handle) != DLT_EN10MB)
47      {
48          fprintf(stderr, "%s n'est pas un réseau ethernet\n", dev);
49          exit(EXIT_FAILURE);
50      }
51
52      /* Définition de la fontion de rappel */
53       pcap_loop(handle, num_packets, Capture::got_packet, NULL);
54
55      printf("\nCapture complete.\n");
56  }
```

Llligne 26 : méthode capture_paquets() de la classe capture.

Ligne 32,33,34 : Affichage du périphérique ou l'on capture, du filtre et du nombre de paquets.

Ligne 37: pcap_open_live capture sur une interface disponible ou préalablement défini, ici dev est l'interface.

Ligne 53 : pcap_loop appelle notre fonction de rappel got_packet passé en argument pour qu'elle l'utilise sans qu'elle ne la connaisse auparavant.

9.4.4 Filtrage UDP

```
58 void Capture::filtre_udp()
59 {
60
61     char filter_exp[] = "udp";      /* expression filtre  */
62
63     /* compile l'expression du filtre */
64     if (pcap_compile(handle, &fp, filter_exp, 0, net) == -1)
65     {
66         fprintf(stderr, "Impossible de compiler le filtre' %s: %s\n",
67                 filter_exp, pcap_geterr(handle));
68         exit(EXIT_FAILURE);
69     }
70
71     /* application du filtre*/
72     if (pcap_setfilter(handle, &fp) == -1)
73     {
74         fprintf(stderr, "Impossible d'installer le filtre' %s: %s\n",
75                 filter_exp, pcap_geterr(handle));
76         exit(EXIT_FAILURE);
77     }
78     |  printf("\n Filtre appliqué\n");
79 }
```

9.4.5 Filtrage SIP

```
81  void Capture::filtre_sip()
82  {
83
84      if(port_src == 5060 || port_dest == 5060)
85      {
86          printf("Packet numero : %s :", compteur);
87          printf("Paquet de protocole SIP");
88      }
89
90  }
```

Ligne 81 : méthode filtre_sip de la classe capture, elle va nous permettre de ne garder que les paquets sip

Ligne 84 : On crée une boucle on l'on ne va y entrer que si les paquets capturés sont des paquets avec le port 5060, soit des paquets SIP

Ligne 86,87 : L'affichage nous permet pour le moment de vérifier le bon filtrage des paquets.

9.4.6 Filtrage RTP & RTCP

```
92  void Capture::filtre_RTP()
93  {
94      if(port_src == port_rtp || port_dest == port_rtp)
95      {
96          printf("Paquet de protocole RTP");
97      }
98
99      if(port_src == (port_rtp + 1) || port_dest == (port_rtp + 1))
100     {
101         printf("Paquet de protocole RTCP");
102     }
103
104 }
```

Ligne 92 : méthode filtre_RTP() elle va nous permettre de ne garder que les paquets RTP car ce sont eux que l'on va décoder pour récupérer les informations de Qualité de service.

Ligne 94 : Boucle où l'on ne passe que lorsque ce sont des paquets RTP soit des paquets avec le port RTP récupéré dans les paquets SIP.

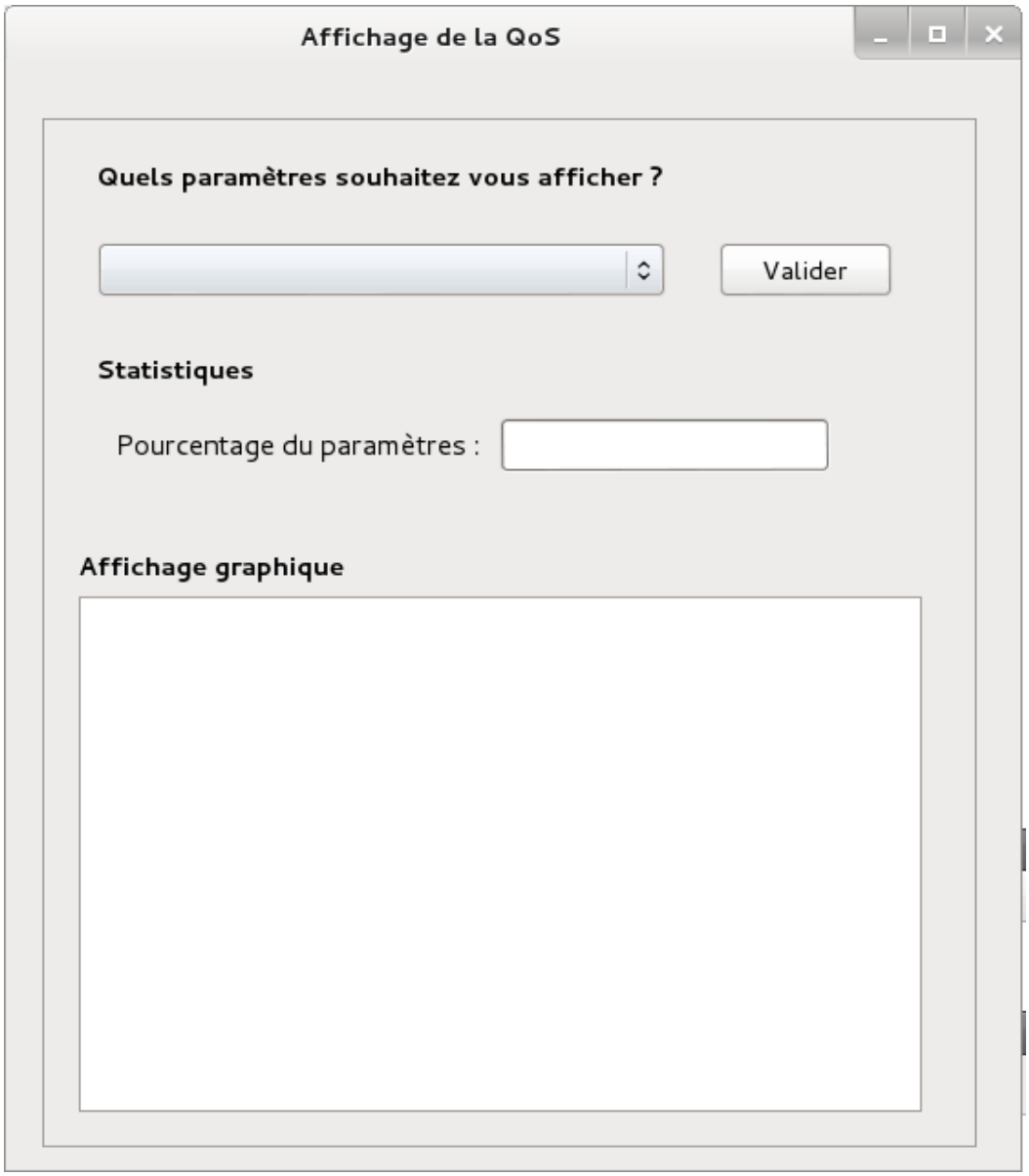
Ligne 96 : Affichage permettant de vérifier le bon filtrage de RTP

Ligne 99 : Boucle où l'on ne passe que lorsque ce sont des paquets RTCP soit des paquets avec le port RTP récupéré dans les paquets SIP + 1.

Ligne 101 : Affichage permettant de vérifier le bon filtrage de RTCP

9.5 Affichage de statistiques

L'affichage des statistiques se fera sur une IHM, une fois la campagne d'appels terminé, une IHM apparaîtra afin de signaler à l'utilisateur la fin de la campagne mais aussi pour qu'il puisse voir et sauvegarder les paramètres de la QoS. L'utilisateur choisira dans une liste déroulante quel paramètre il souhaite afficher. Il verra ensuite la valeur moyenne de se paramètre sur l'ensemble de la communication ainsi que son évolution au cours de celle-ci



9.6 Enregistrement des statistiques

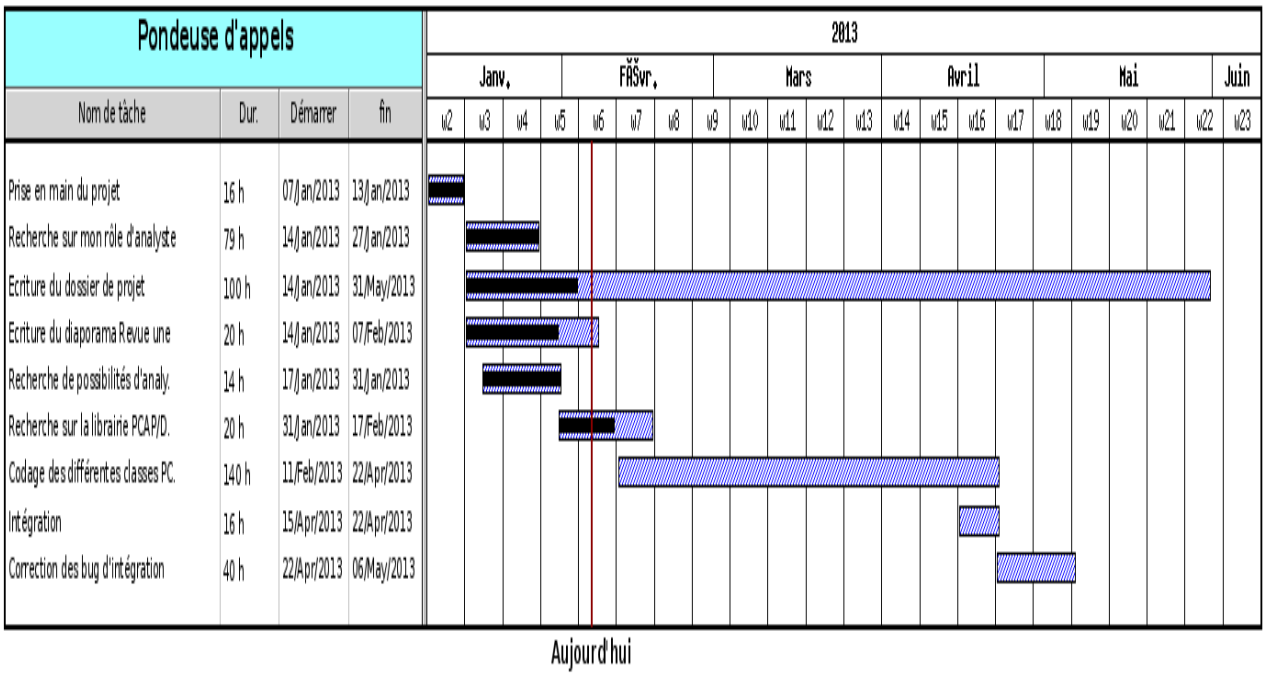
L'enregistrement des statistiques se fera dans un fichier XML, que QT pourra rouvrir afin de le relire.

Deux approches s'offrent à nous pour générer des fichiers XML à partir d'applications Qt :

- générer un arbre DOM et appeler save() sur celui ci ;
- générer le code XML manuellement.

Le choix entres ces approches est souvent indépendant du fait que nous utilisions SAX ou DOM pour la lecture des documents XML.

9.7 Conclusion


Pondeuse d'appels
2013
Janv.
Févr.
Mars
Avril
Mai
Juin
Nom de tâche
Dur.
Démarrer
fin
Prise en main du projet 16 h 07/Jan/2013 13/Jan/2013
Recherche sur mon rôle d'analyste 79 h 14/Jan/2013 27/Jan/2013
Ecriture du dossier de projet 100 h 14/Jan/2013 31/May/2013
Ecriture du diaporama Revue une 20 h 14/Jan/2013 07/Feb/2013
Recherche de possibilités d'analy. 14 h 17/Jan/2013 31/Jan/2013
Recherche sur la librairie PCAP/D. 20 h 31/Jan/2013 17/Feb/2013
Codage des différentes classes PC. 140 h 11/Feb/2013 22/Apr/2013
Intégration 16 h 15/Apr/2013 22/Apr/2013
Correction des bug d'intégration 40 h 22/Apr/2013 06/May/2013
Aujourd'hui

Ce projet m'a permis de m'investir dans les technologies d'actualités tout en m'adaptant au cahier des charges. Tout au long de celui-ci, j'ai donc pu apprendre toute la démarche à mettre en place sur un projet concret : de la lecture du cahier des charges à l'écriture de notre démarche de développement en passant par le développement et donc la conception de l'application demandée.

Il m'a beaucoup apporté au plan technique par la mise en pratique des connaissances acquises en BTS IRIS. J'ai appris à développer mes connaissances à l'aide de recherches et d'investissements sur ce projet. Il m'a permis aussi de m'intégrer et apprendre à travailler en groupe et de mieux comprendre la réalisation d'un projet.

L'ambiance de groupe fut toujours positive, une aide mutuelles et un respect de chacun ont grandement aidé à la finalisation de ce projet demandé par les MMA.

Pour ma part au moment de l'écriture, l'enregistrement dans un fichier des résultats de QoS n'est pas opérationnel j'ai donc un retard sur mon planning prévisionnel et que je dois rattraper avant la fin du projet.

10. Fiches de test : Prieur Quentin

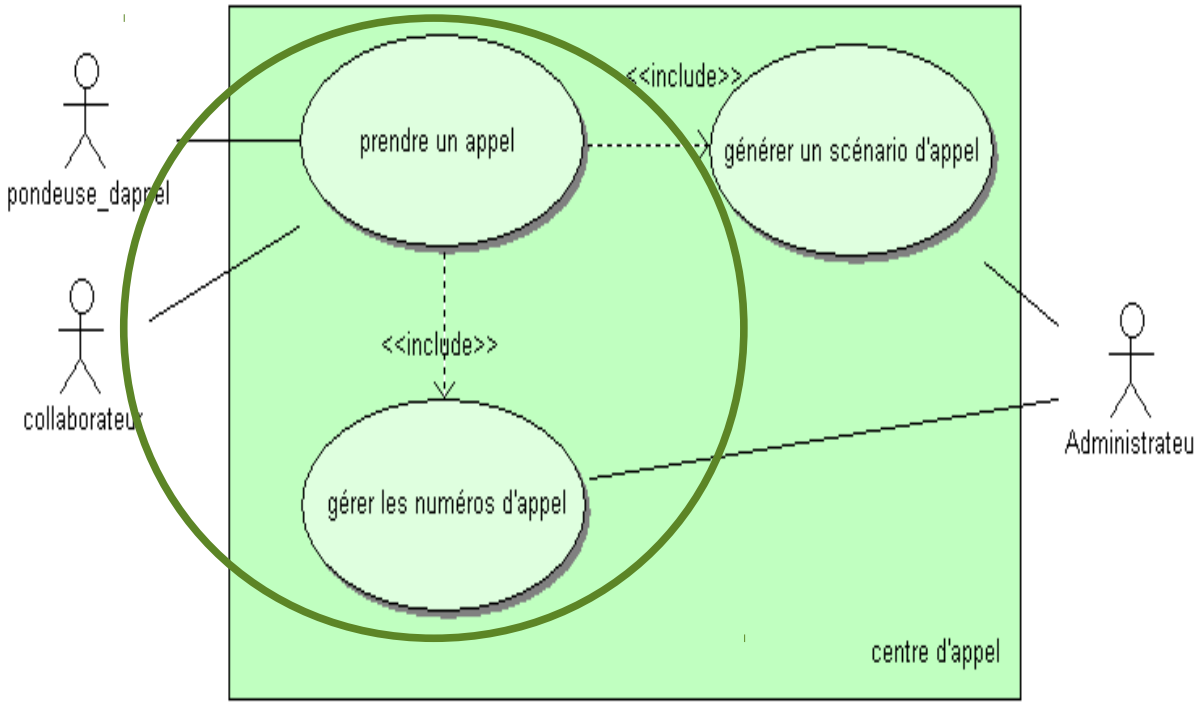
10.1 Plan de test

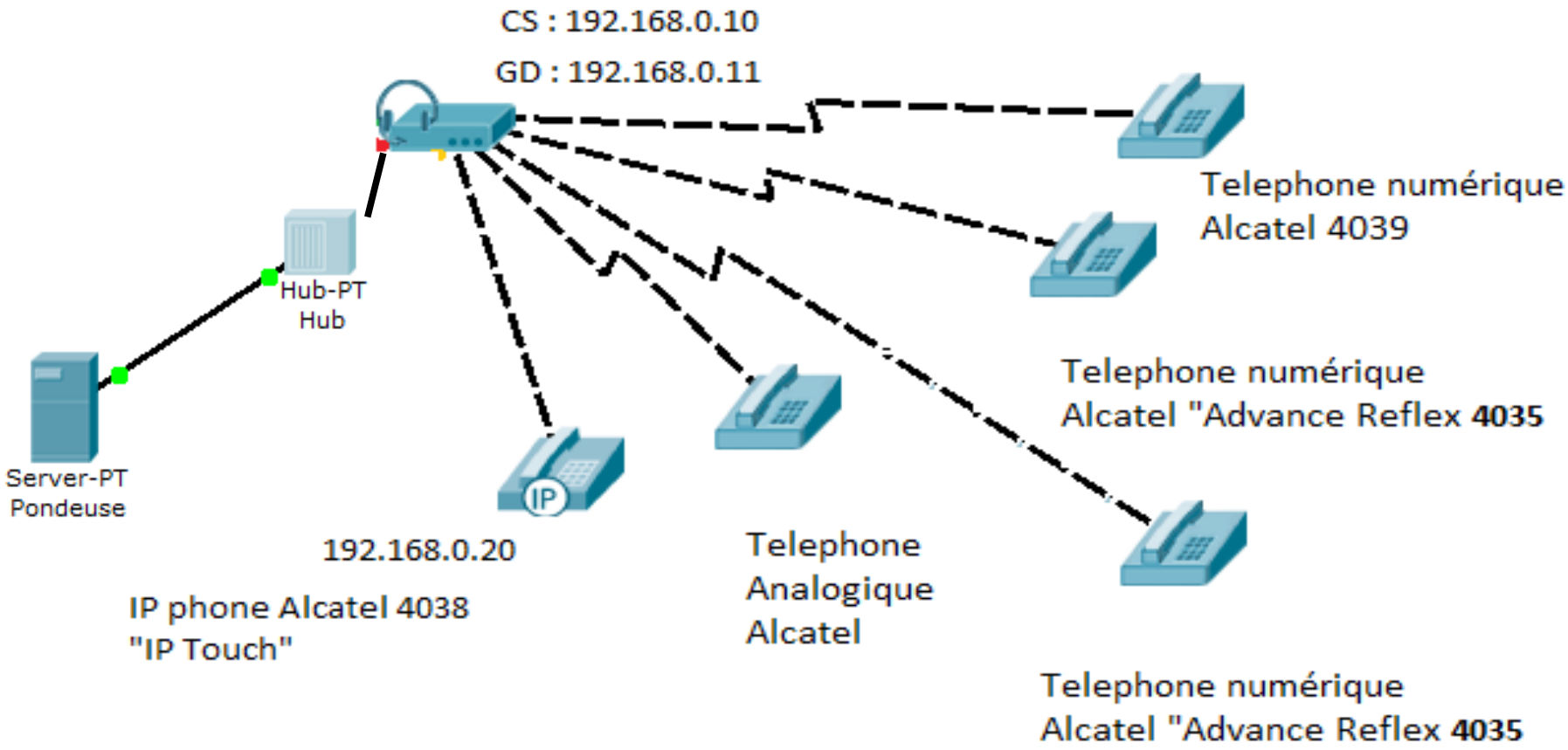
Installation d'un poste sur l'OXE Alcatel Test Fonctionnel

Mise en place d'un poste de type numérique	
Date du test:	07/05/2013
Type du test:	Fonctionnel
Objectifs du test:	1. Réaliser l'installation d'un poste de type 4035 2. Création et assignation d'un usager 3. Modification d'un branchement téléphonique
Conditions du test	
État initial	**Environnement du test**
• L'OXE Alactel est sous tension et correctement initialisé. • Le pc de contrôle ayant pour ip fixe 192.168.0.50 est directement connecté sur le hub (concentrateur) représenté par la carte LANX8-2	• Se connecter via le logiciel putty au call server : 192.168.0.10:23 User : mtcl / Password : mtcl • Le pc a accès au menue de gestion via la commande mgr • Le téléphone doit être de type 4035T et doit être connecter a un port numérique non configuré.
Procédures du test	
Opérations	**Résultats attendus**
TEST n°1 Se déplacer dans le menu : Usager >> Création Renseigner les champs suivants : No d'annuaire : 1007 Nom d'annuaire : TEST Prénom : REVUE Type de poste : 4035T Code secret : 0000 No poste associé : 1008 poste ACD : poste Pro ACD Tous les autres paramètres doivent rester au réglages par défauts. Valider avec *CTRL+V* puis une seconde fois *CTRL+V*	Le poste doit exister dans la liste. Pour vérifier il suffit de se déplacer dans,consultation/modification. Sélectionner toutes les instances et valider avec *CTRL+V.* Le poste doit être présenter dans la liste .

TEST n°2 Procédure d'assignation : - décrocher le combiné - composer le numéro du poste:1007 - code personnel : 0000 puis raccrocher le combiné. Le téléphone sonne à nouveau décroché et raccroché.	Le téléphone est assigné à son numéro d'appel, les appels vers le numéro défini seront normalement redirigés vers le poste.
TEST n°3 - Situer la le connecteur qui relie le téléphone 4035T a la carte Digital interface - Débrancher le connecteur et le rebrancher sur un port « Digital Interface »	Le téléphone doit fonctionner, comme auparavant sur le port RJ45 précédent.

Situation dans le cas d'utilisation :




CS : 192.168.0.10
GD : 192.168.0.11
Telephone numérique
Alcatel 4039
Hub-PT
Hub
Telephone numérique
Alcatel "Advance Reflex 4035
Server-PT
Pondeuse
IP
192.168.0.20
IP phone Alcatel 4038
"IP Touch"
Telephone
Analogique
Alcatel
Telephone numérique
Alcatel "Advance Reflex 4035

10.2 Procès verbal de test

Installation d'un poste sur l'OXE Alcatel
Test Fonctionnel

Mise en place d'un poste de type numérique		
Date du test:	18/04/2013	
Type du test:	Fonctionnel	
Objectifs du test:	1. Réaliser l'installation d'un poste de type 4035 2. Création et assignation d'un usager 3. Modification d'un branchement téléphonique	
Conditions du test		
État initial	**Environnement du test**	
• L'OXE Alactel est sous tension et correctement initialisé. • Le pc de contrôle ayant pour ip fixe 192.168.0.50 est directement connecté sur le hub (concentrateur) représenté par la carte LANX8-2	• Se connecter via le logiciel putty au call server : 192.168.0.10:23 User : mtcl / Password : mtcl • Le pc a accès au menue de gestion via la commande mgr • Le téléphone doit être de type 4035T et doit être connecter a un port numérique non configuré.	
Procédures du test		
Opérations	**Résultats attendus**	**Résultats obtenues**
TEST n°1 Se déplacer dans le menu : Usager >> Création Renseigner les champs suivants : No d'annuaire : 1007 Nom d'annuaire : TEST Prénom : REVUE Type de poste : 4035T Code secret : 0000 No poste associé : 1007 poste ACD : poste Pro ACD Tous les autres paramètres doivent rester au réglages par défauts. Valider avec *CTRL+V* puis une seconde fois *CTRL+V*	Le poste doit exister dans la liste. Pour vérifier il suffit de se déplacer dans, consultation/modification. Sélectionner toutes les instances et valider avec *CTRL+V.* Le poste doit être présenter dans la liste .	**Conforme**

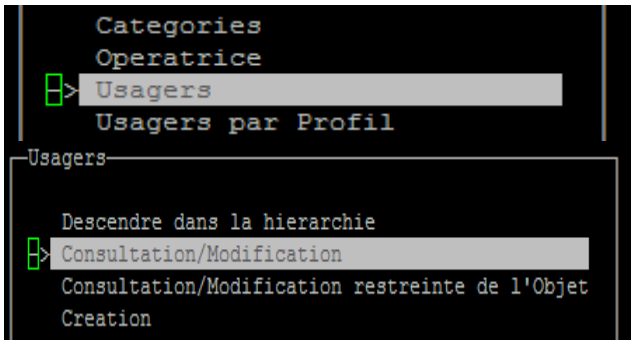
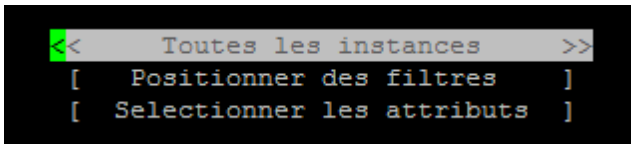
TEST n°2 Procédure d'assignation : - décrocher le combiné - composer le numéro du poste:1007 - code personnel : 0000 puis raccrocher le combiné. Le téléphone sonne à nouveau décroché et raccroché.	Le téléphone est assigné à son numéro d'appel, les appels vers le numéro défini seront normalement redirigés vers le poste.	**Conforme**
TEST n°3 - Situer la le connecteur qui relie le téléphone 4035T a la carte Digital interface - Débrancher le connecteur et le rebrancher sur un port « Digital Interface »	Le téléphone doit fonctionner, comme auparavant sur le port RJ45 précédent.	**Non conforme** Le téléphone demande le numéro, mais sonne occupé, signe d'un dysfonctionnement dans la procédure.

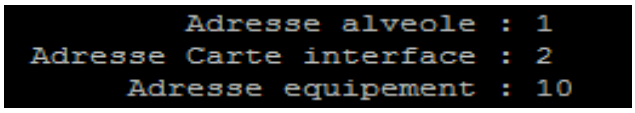
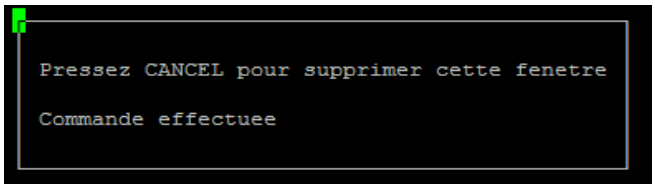
10.3 Plan de test structurel

Changement de connecteur d'un poste de type 4035

Modification du branchement d'un poste de type 4035	
Date du test:	07/05/2013
Type du test:	Structurel
Objectifs du test:	1. Réaliser le changement de connecteur d'un poste de type 4035

Conditions du test	
État initial	**Environnement du test**
• L'OXE Alactel est sous tension et correctement initialisé. • Le pc de contrôle ayant pour ip fixe 192.168.0.50 est directement connecté sur le hub (concentrateur) représenté par la carte LANX8-2	• Se connecter via le logiciel putty au call server : 192.168.0.10:23 User : mtcl / Password : mtcl • Le pc a accès au menu de gestion via la commande mgr

Procédures du test	
Opérations	**Capture d'écran**
Opération 1 : Via la menu *mgr* se déplacer dans *Usagers* puis dans *Consultation / Modification.*	
Sélectionner *Toutes les instances* avec la touche *Entrer* puis valider avec *CTRL+V*	

Opération 2 : Sélectionner le poste : 1007 (en appuyant sur *Entrer)* Trois paramètres sont à modifier : *Adresse alvéole : 255* *Adresse Carte interface : 255* *Adresse équipement : 255* Valider les modifications avec *CTRL+V*	 

Opération 3 :

Refaire la procédure d'assignation :
- décrocher le combiné
- composer le numéro du poste:1007
- code personnel : 0000

puis raccrocher le combiné.

Le téléphone sonne à nouveau décroché et raccroché.

11. Fiche de test : Plateau Mathias

Mise a jour de la base de données via l'IHM	
Date du test:	18/04/2013
Type du test:	Fonctionnel
Objectifs du test	Vérifier l'ajout d'un nom de campagne, d'un numéro d'appel et d'un temps d'appel dans la base de données grâce à la méthode on_pushButtonAppeler_clicked() de la classe Campagne.

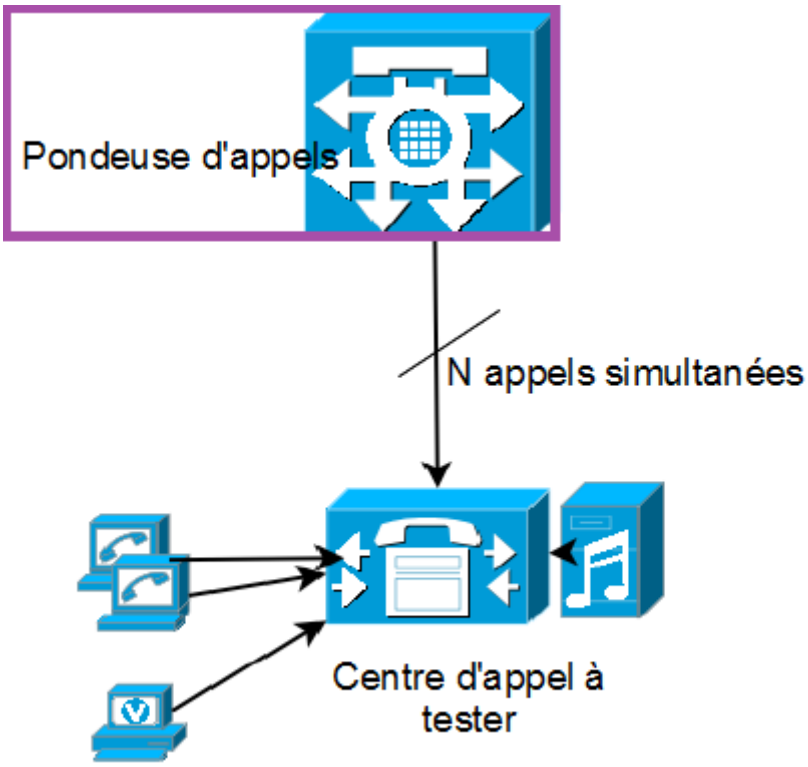
Conditions du test	
État Initial	**Environnement du test**
- Le serveur base de données MYSQL situé sur la trixbox est en marche, adresse ip : 172.17.83.170. -Tables utilisées pour le test : campagne. - L'utilisateur a lancé l'éxécutable « MmaIHM ».	-Qt Creator -MYSQL -phpMyAdmin -Projet compilé en mode debug

Procédures du test		
Opérations	**Résultats attendus**	**Résultats obtenus**
TEST n°1 *Interaction avec la base de données : ajout d'un champ quelconque.* *Cas n°1 : la base de données ne contient aucun* nom de campagne, temps d'appel et numéro d'appels dans les champs nom_campagne, temps_appel et numero_appel de la table campagne.	➢ L'utilisateur vérifie que la base de données ne contient aucun nom de campagne, temps d'appel et numéro d'appels dans les champs nom_campagne, temps_appel et numero_appel de la table campagne.	**Ok**
Cas n°2 : *-Champ Nom de campagne rempli sur la fenêtre configuration,* *-Appui sur le bouton « Suivant »* *-Champ numéro d'appels et temps d'appel remplis sur la fenêtre Campagne.* *-Appui sur le bouton Appeler*	➢ Le nom de campagne est alors enregistré dans la table « campagne » ,dans le champ « nom_campagne ». Le « numéro d'appel » et le « temps d'appel » sont enregistrer dans les champs « temps_appe l » et « numero_appel » de la table campagne de la base de données MYSQL.	**Ok**

TEST n°2 *Ajout d'un champ vide pour le nom de campagne.*	➢ Lors de l'appui sur le bouton suivant, le programme demande à l'utilisateur de rentrer un nom de campagne valide dans la fenêtre.	**Ok**
TEST n°3 *Interaction avec la base de données : ajout d'un champ de campagne déjà présent dans la table de la base de données.*	➢ Lors de l'appui sur le bouton suivant, le programme prévient l'utilisateur grâce une fenêtre d'avertissement que le nom de campagne fournit est déjà utilisé pour une autre campagne.	**Ok**
TEST n°4 *Interaction avec la base de données : ajout d'un champ similaire grâce aux majuscules et minuscules à celui qui se trouve déjà dans la base, exemple : « Test » et « test ».*	➢ Le champ est reconnu comme similaire à celui déjà présent, le programme prévient l'utilisateur grâce une fenêtre d'avertissement que le nom de campagne fournit est déjà utilisé pour une autre campagne. Nous pouvons en conclure que « test=Test ».	**Ok**
TEST n°5 *Ajout d'un champ vide pour le numéro d'appel.*	➢ Lors de l'appui sur le bouton Appeler, le programme demande à l'utilisateur de rentrer un numéro d'appels valide dans la fenêtre.	**Ok**
TEST n°6 *Délimitation du temps d'appel*	➢ L'utilisateur peut uniquement saisir un temps d'appel entre 1 et 1800s.	**Ok**

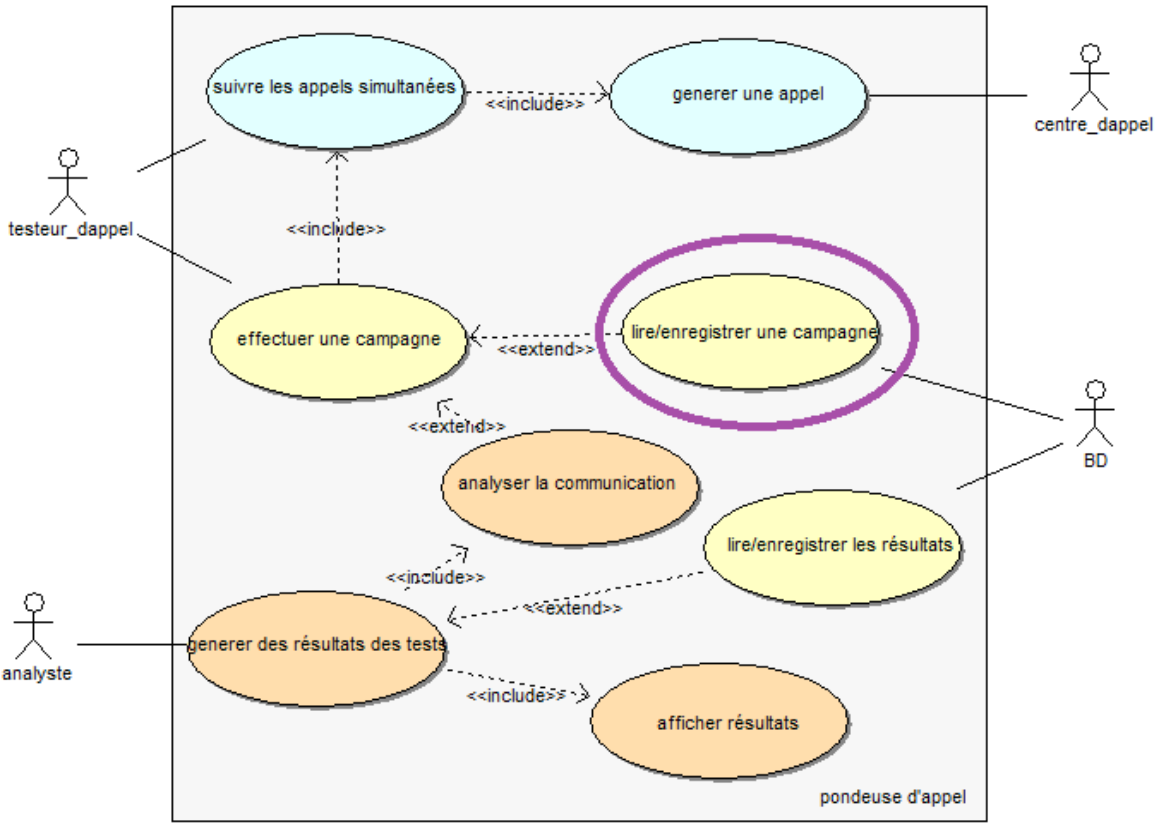
Niveau du test :

Nous nous situons Ici



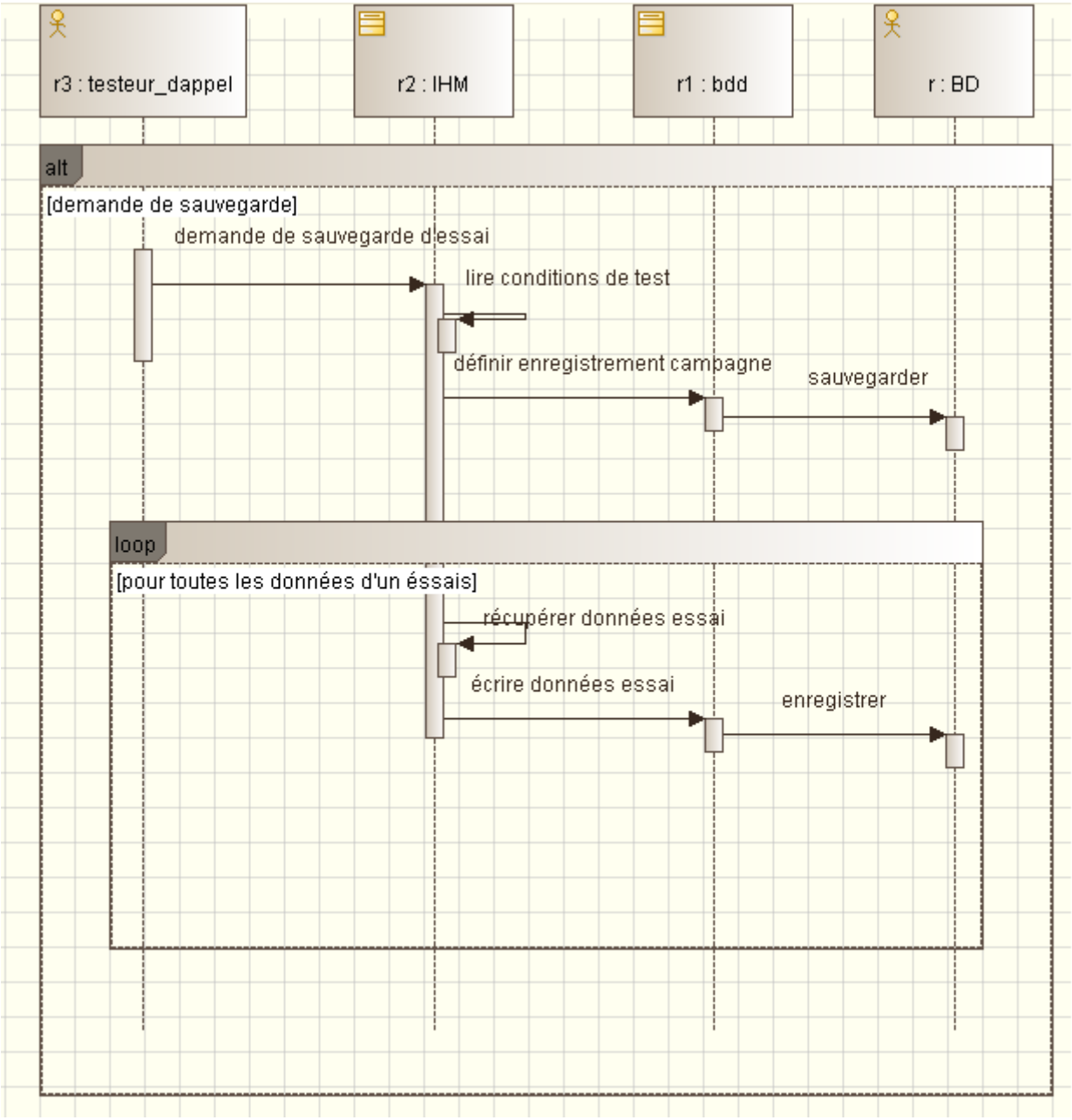
Uniquement du côté de la pondeuse d'appels.

Cas d'utilisation enregistrer une campagne.



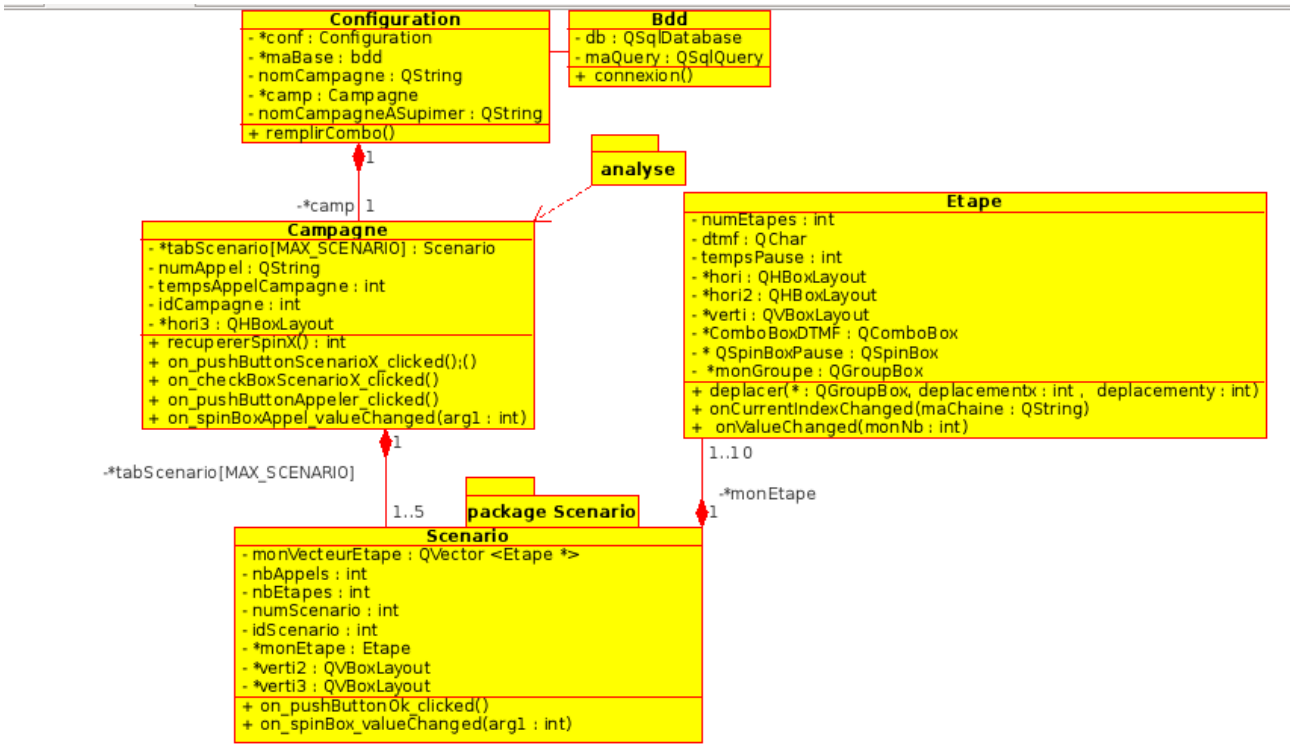
Pour ce test, nous nous situons dans le cas d'utilisation enregistrer une campagne, la base de données en est donc un acteur principal car les données seront stockées à l'intérieur.

Procédure d'enregistrement :



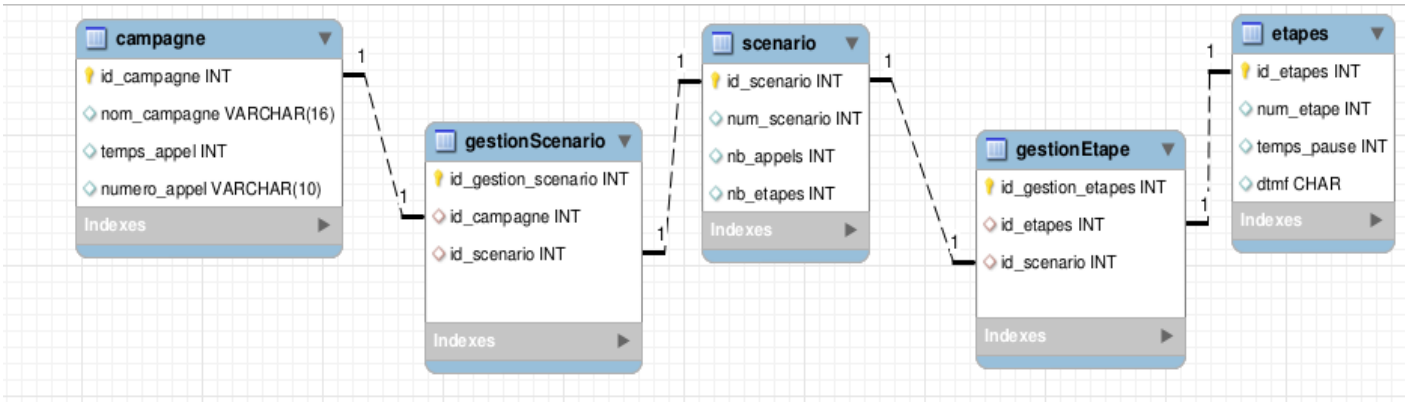
Voici le diagramme de séquence auquel nous pouvons nous référer pour ce test, nous allons donc demander un accès à la base de données, récupérer les données à partir de l'IHM puis rentrer les données dans la base de données.

Diagramme de classes :



Nous nous servirons de la classe Configuration, Campagne et bdd pour ce test. La classe Configuration contient l'attribut nomCampagne de type Qstring que nous devons mettre dans la base de données mais aussi de la classe bdd afin de pouvoir se servir des requêtes SQL. La classe Campagne contient les attributs tempsAppelCampagne et numAppel.

Tables de la base de données:



Ici, seulement la table campagne servira, en effet la chaîne de caractères que contient l'attribut nomCampagne sera entrée dans le champ nom_campagne , les entiers que contienent les attribut tempsAppelCampagne et numAppel seront entrés dans les champs temps_appel et numero_appel de la table campagne.

12. Fiche de test : Chrétien Clément

Plan de test

Utilisation du protocole FTP et génération de fichier

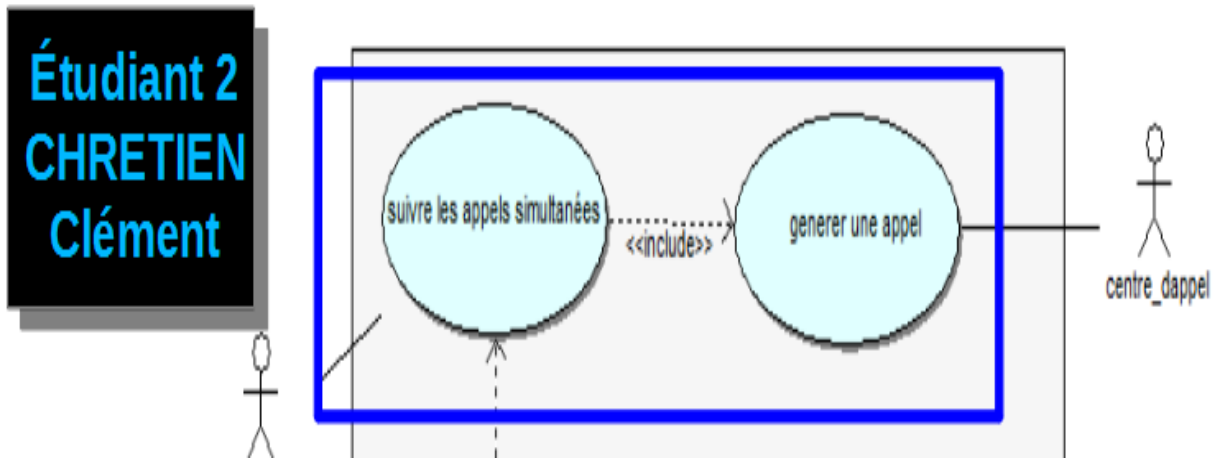
Test fonctionnel: générer un appel sur la trixbox

Utilisation du protocole FTP et génération de fichier	
Date du test:	18/04/13
Type du test:	Fonctionnel
Objectifs du test:	1. Vérifier et valider le fonctionnement de la méthode d'envoi du fichier qui se base sur le protocole FTP: Classe ClientFTP Méthode : envoyerFichier(QString chemin) 2. Test de la génération de fichier Classe : Appel Méthode:genererFichier()

Conditions du test	
État initial	**Environnement du test**
• L'application est présente sur le pc client (Windows ou Linux). • Le pc client est relié par un câble Ethernet sur le réseau de la trixbox • les paramètres du protocole FTP sont configurer dans le registre du système d'exploitation grâce à l'application prévue à cet effet	• Le test du résultat des méthodes se fera à l'aide de la classe QtestLib de QT. • L'envoi du fichier pourra être visible sur la trixbox. • L'application utilisera une interface graphique pour lancer les méthodes de l'application.

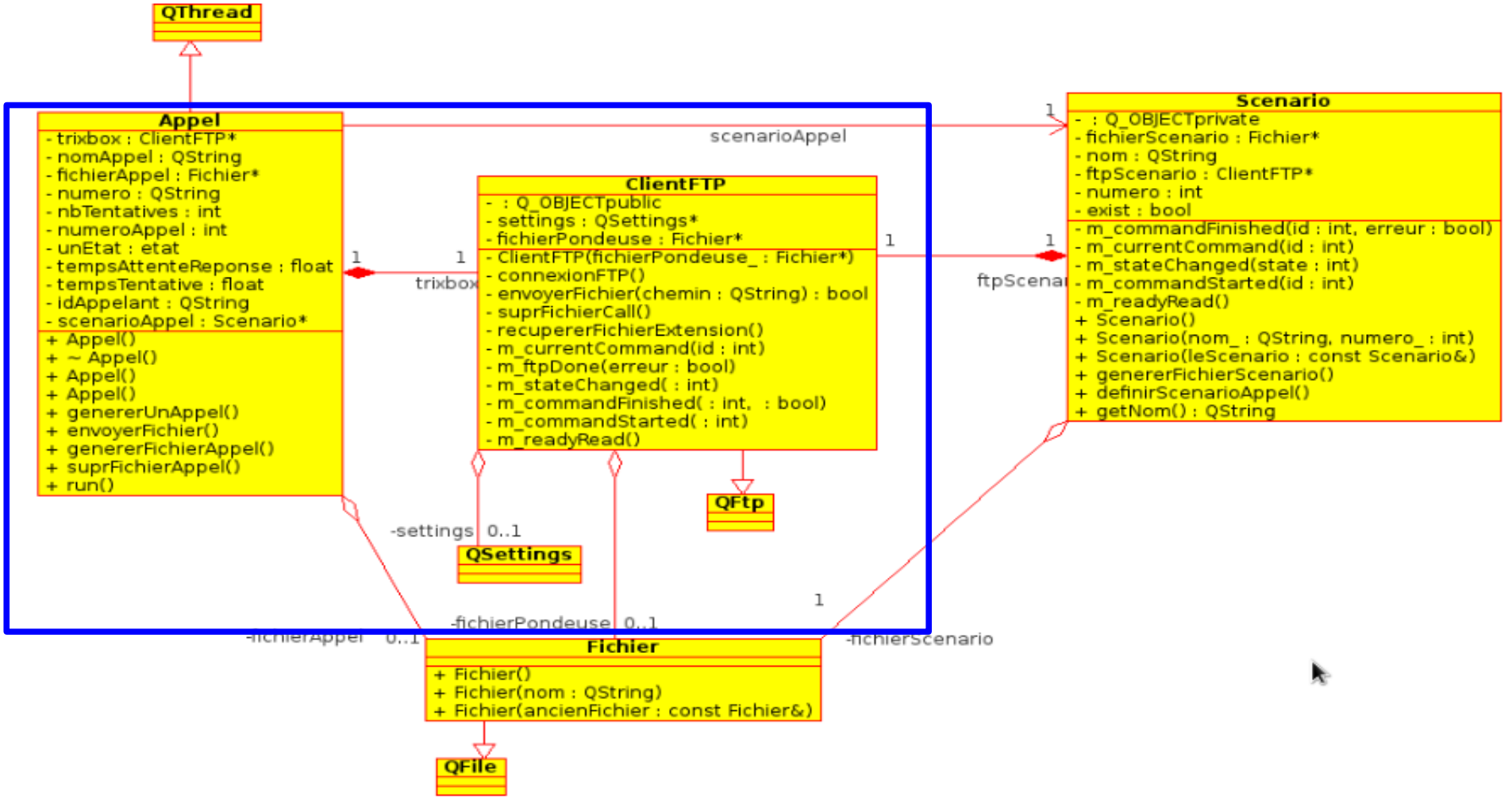
Procédures du test	
Opérations	**Résultats attendus**
TEST n°1 - Vérifier que le répertoire dû ne contient aucun fichier. - Lancement du test avec QtestLib.	➢ Le fichier est présent dans le répertoire du projet ➢ le fichier contient : ◦ Channel: SIP/10 ◦ WaitTimes: 5 ◦ RetryTimes: 2 ◦ MaxRetries: 5 ◦ Context: test ◦ Extension: s ◦ CallerID: clement ➢ le test est terminé avec succès

TEST n°2 - Vérifier que le répertoire de destination du FTP **/var/spool/asterisk/outgoing** ne contient pas déjà le fichier. Commande « ls -l -a » -lancement du test de l'envoi avec QT creator	➢ La méthode n'a pas renvoyé false. ➢ Le fichier est présent sur le serveur, ➢ l'appel se déclenche



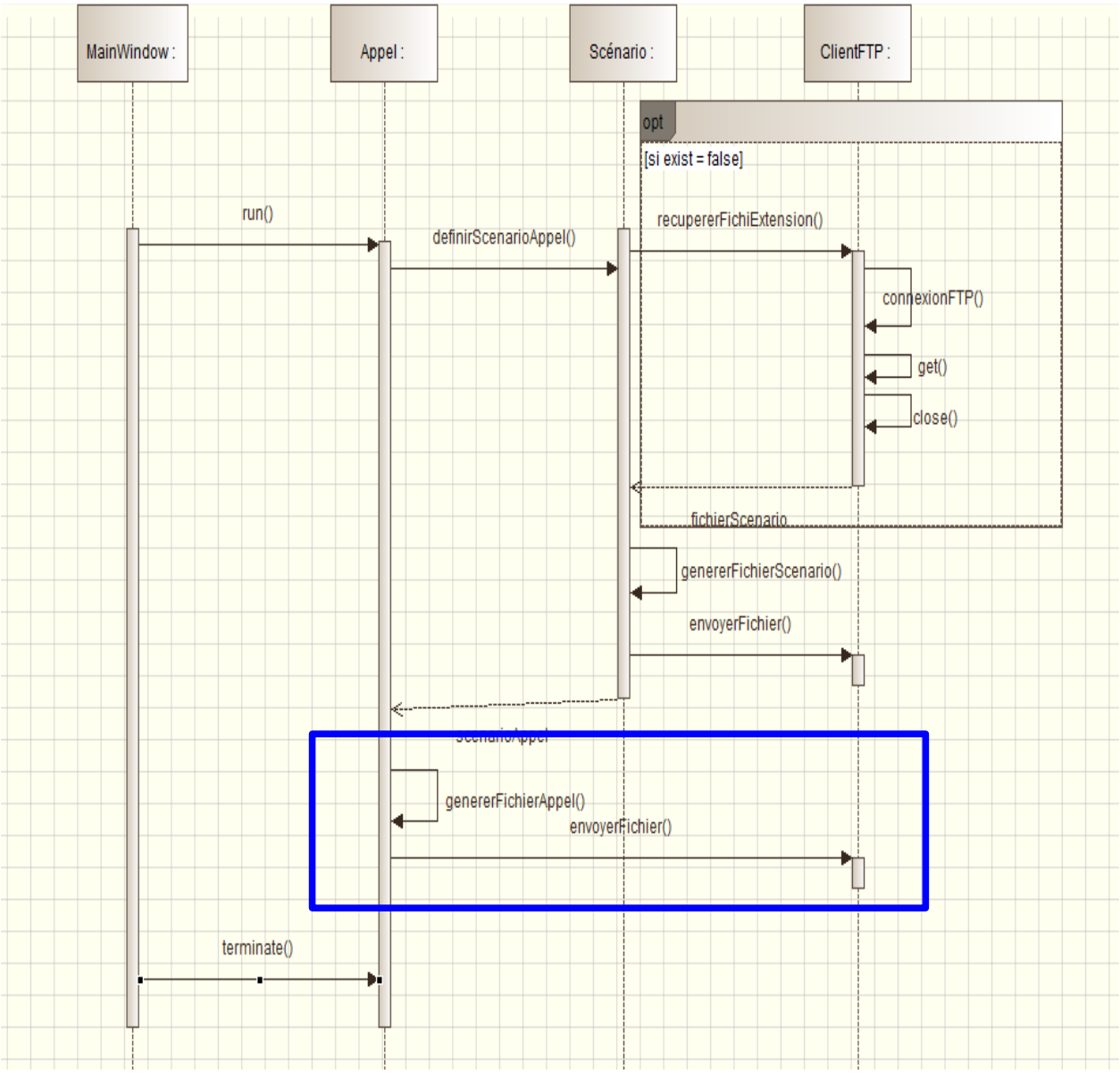
Les cas d'utilisation qui me concernent sont ceux qui sont entourés sur l'image ci-dessus.
Le Cas d'utilisation qui sera testée lors de ce test est « générer un appel.

Diagramme de Classe :

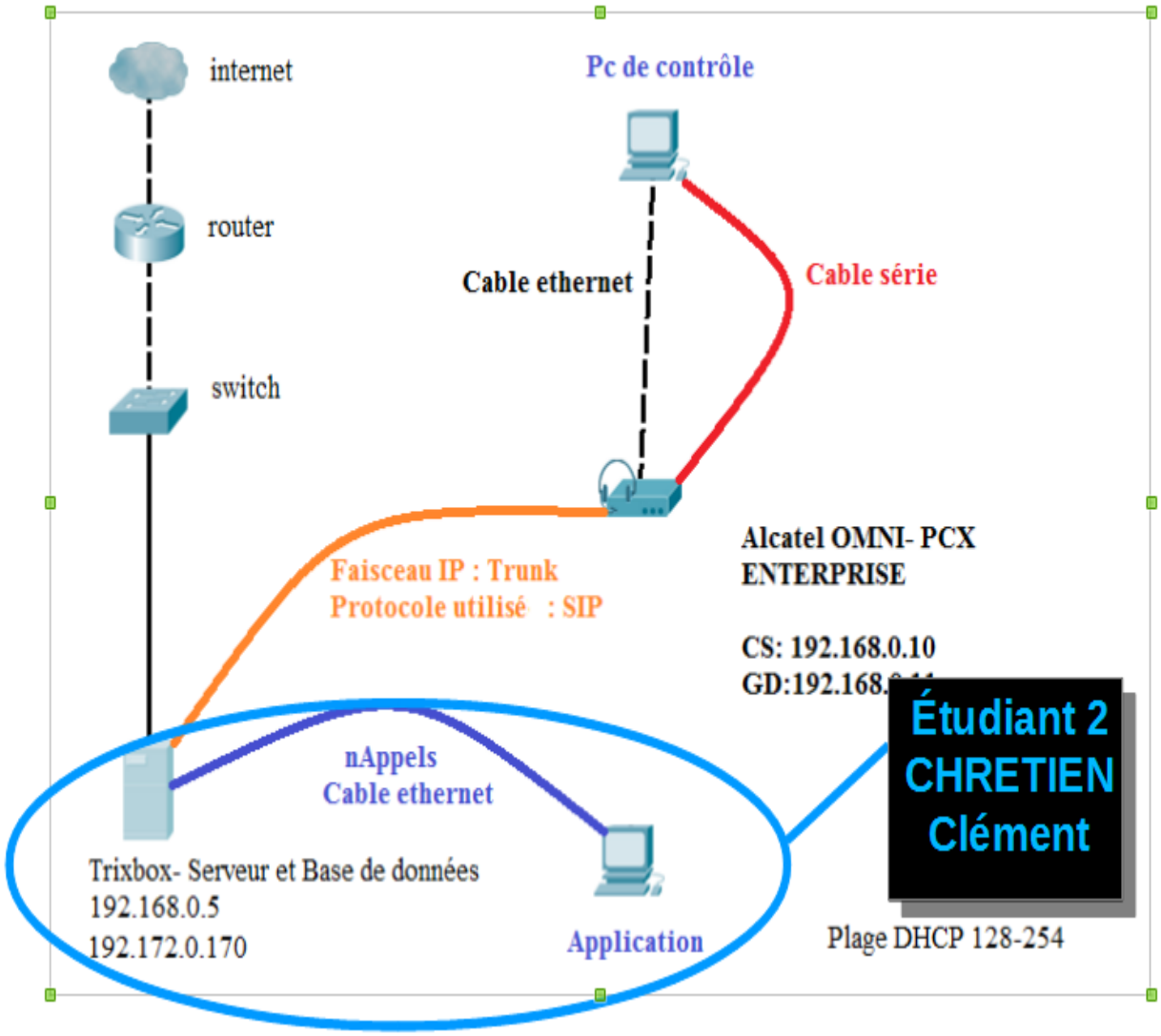


Les classes qui seront testées sont les classes encadrées en bleu.

Voici la partie de la génération d'appel qui sera testé. Elle concerne la génération du fichier d'appel et l'envoi de ce fichier sur la trixbox à l'aide du protocole FTP.

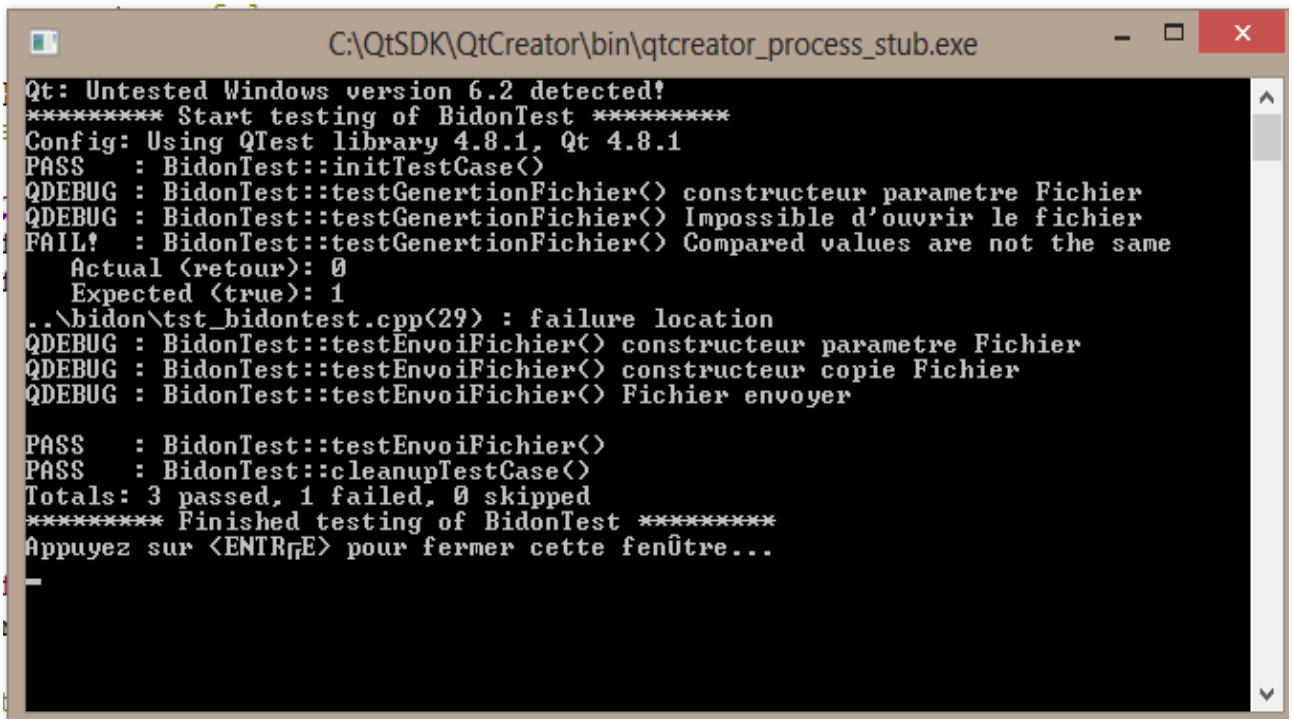


Architecture réseau du projet :



12.1.1 Test n°1 :

le test numéro 1 n'a pas donné les résultats attendus à l'origine qui étaient la création du fichier permettant l'appel :


C:\QtSDK\QtCreator\bin\qtcreator_process_stub.exe
Qt: Untested Windows version 6.2 detected!
********* Start testing of BidonTest *********
Config: Using QTest library 4.8.1, Qt 4.8.1
PASS : BidonTest::initTestCase()
QDEBUG : BidonTest::testGenertionFichier() constructeur parametre Fichier
QDEBUG : BidonTest::testGenertionFichier() Impossible d'ouvrir le fichier
FAIL! : BidonTest::testGenertionFichier() Compared values are not the same
Actual (retour): 0
Expected (true): 1
..\bidon\tst_bidontest.cpp(29) : failure location
QDEBUG : BidonTest::testEnvoiFichier() constructeur parametre Fichier
QDEBUG : BidonTest::testEnvoiFichier() constructeur copie Fichier
QDEBUG : BidonTest::testEnvoiFichier() Fichier envoyer
PASS : BidonTest::testEnvoiFichier()
PASS : BidonTest::cleanupTestCase()
Totals: 3 passed, 1 failed, 0 skipped
********* Finished testing of BidonTest *********
Appuyez sur <ENTRﻦE> pour fermer cette fenûtre...

Comme nous pouvons voir sur l'image située au-dessus un des tests n'a pas été réussi et si on regarde le test de la génération du fichier on peut voir que l'on passe dans le constructeur paramétré de la classe Fichier. Ensuite on peut voir un message qui nous dit que le programme n'a pas pu ouvrir le fichier « Impossible d'ouvrir le fichier» et si on se reporte à la méthode qui est testée on peut voir ceci :

```
{
    bool retour;
    if(!fichierAppel->open(QIODevice::ReadOnly | QIODevice::Text))      //ouverture du fichier
       {
           qDebug()<<"Impossible d'ouvrir le fichier";
           retour=false;

       }
    else
    {
       QTextStream flux(fichierAppel);         //création d'un flux pour écrire dans le fichie
       flux.setCodec("UTF-8");          //codec utilisé
       flux <<"Channel: SIP/"<<numero<<endl<<           //ecriture dans le fichier
             "WaitTimes: "<<tempsAttenteReponse<<endl<<
             "RetryTimes: "<<tempsTentative<<endl<<
             "MaxRetries: "<<nbTentatives<<endl<<
             "Context: test"<<endl<<
             "Extension: s"<<endl<<
```

On peut voir que le message « Impossible d'ouvrir le fichier» se déclenche que si le fichier ne peut pas être ouvert en lecture or la méthode écrit dans le fichier. Donc ce test m'a permis de voir une erreur dans ce code.

12.1.2 Test n°2 :

Le test numéro montre l'envoie du fichier sur la trixbox ainsi que son exécution qui est la génération de l'appel.

Le test numéro 2 s'est passé comme il était prévu :

- vérification du répertoire de destination → répertoire vide
- lancement de la méthode → envoie du fichier
- exécution de l'appel → le téléphone à sonné
- fin du test → le fichier a été supprimer après son exécution

13. Fiche de test : Dauvergne Florian

Plan de test
Test des méthodes capture_paquets et filtre_udp de la classe Capture

Test des fonctions de capture et de filtre	
Date du test:	18/04/13
Type du test:	Test fonctionnel
Objectifs du test:	1. Tester capture_paquets 2. Tester le filtrage des paquets

Conditions du test

État Initial	**Environnement du test**
• L'application est présente sur le pc de test fonctionnant sous linux debian • Le pc de test est relié par un câble Ethernet sur la carte LANX8-2 jouant le rôle de hub. • Les commandes de capture de paquets seront lancées via le terminal du pc de test.	• Capture un nombre de paquets prédéfinis auparavant • Capture sur l'interface choisie • Filtre les paquets suivant le protocole UDP • Vérification avec wireshark

Procédures du test

Opérations	**Résultats attendus**
TEST n°1 - Lancer une capture d'une dizaine de paquets sur eth0 - Lancement du test avec QtestLib.	➢ Les paquets transités doivent être capturés que nous soyons branchés sur l'oxe directement ou que nous soyons branchés sur le hub relié à l'oxe. ➢ Aucun paquet ne doit être capturés si nous ne sommes pas relié.
TEST n°2 - Filtrer les paquets suivant le protocole défini dans notre application.	➢ Seul les paquets concernant notre filtre doivent être capturés. Par exemple lors de la configuration d'un filtre DNS, seul des paquets UDP doivent être capturés.

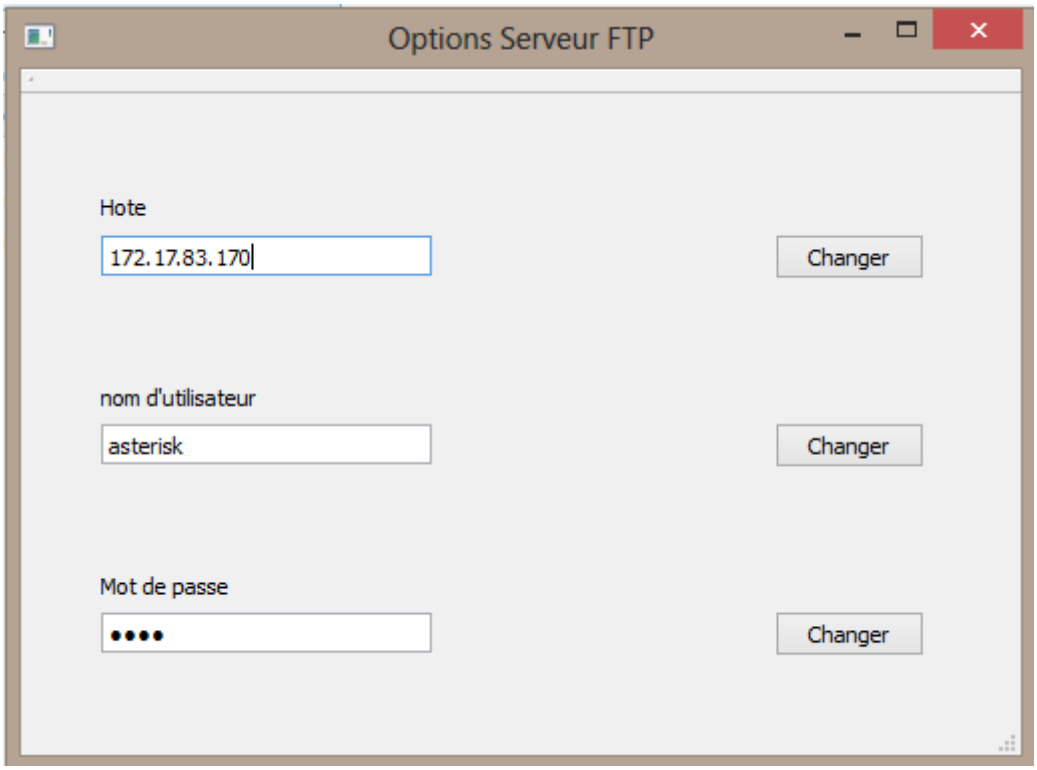
14. Manuel d'utilisation OptionServeur

14.1 Principe de fonctionnement :

Cette application a pour but de sauvegarder les données nécessaires à la connexion au serveur FTP. Elle donne aussi la possibilité de les modifier.

14.2 Présentation :

Voici comment se présente l'application :



elle comporte 3 champs :

- l'adresse IP de l'Hôte
- le nom d'utilisateur
- le mot de passe correspondant

Au lancement de l'application pour la première fois les champs seront vides, car aucune donnée n'est présente, il faudra donc remplir les champs avant la première utilisation de la pondeuse.

14.3 Modifications d'une ou plusieurs données :

Pour la modification de données il faut lancé l'application Option Serveur, à son ouverture les champs seront remplis avec les données présentes actuellement sur le PC. Pour les modifier on édite la ou les zones de texte à modifier et après modification appuyée sur le bouton « Changer » des lignes concernées.

Après tout changement la ligne modifiée est effacée pour valider la modification.

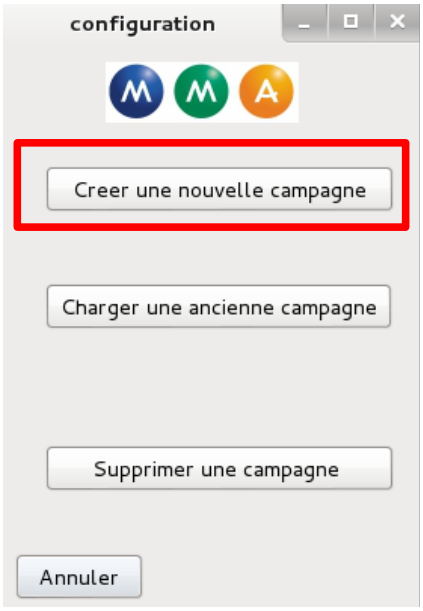
15. Manuel d'utilisation de l'interface homme-machine :

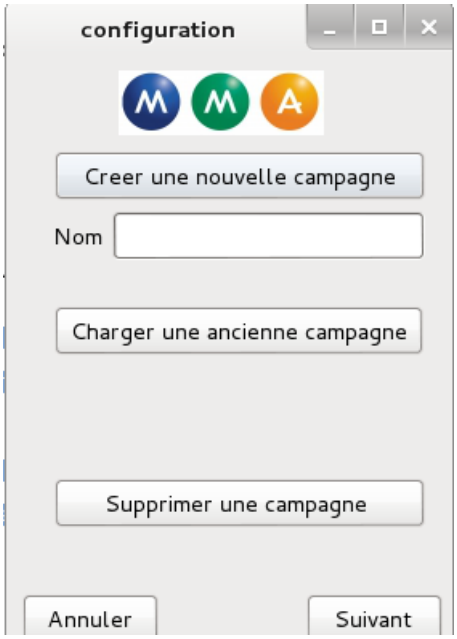
15.1 Nouvelle campagne :

Prérequis: Lancement de l'exécutable, résolution recommandée, 1600x900.

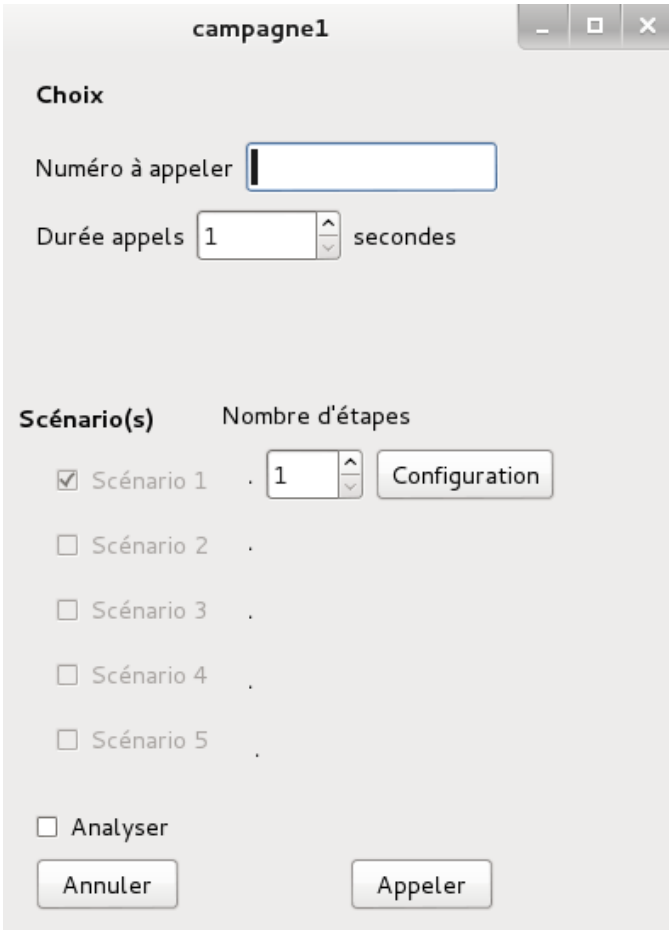
Pour commencer une campagne d'appels, il faut tout d'abord la créée, lancez alors l'exécutable du programme.

Vous arrivez sur une fenêtre de configuration, cliquez alors sur «Créer une nouvelle campagne»:



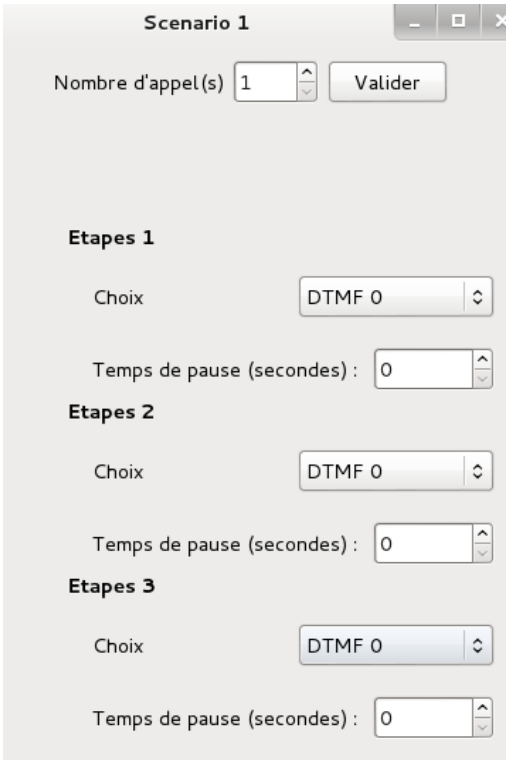


Un champ vous permettant de saisir un nom de campagne apparaît alors, saisissez votre nom de campagne, maximum 16 caractères, sans espaces ni caractères spéciaux. Une fois le nom de campagne saisi, cliquez alors sur le bouton Suivant.
Vous arrivez alors sur une fenêtre pour paramétrer la campagne :



- Entrez le numéro à appeler (uniquement des chiffres, 10 maximum).
- Entrez votre temps d'appel total, par défaut le temps d'appel est de 1 seconde (1800 secondes maximum).
  Attention si vous entrez un temps d'appels inférieur à la somme total des temps de pause de chaque étapes, l'appel finira avant la fin de l'envoi des DTMF.
- Sélectionnez le nombre d'étapes pour un scénario (1 minimum, 10 maximum).
  Par défaut nous avons 1 étape pour le scénario 1 qui envois un DTMF0.

- Cliquez ensuite sur le bouton du scénario pour le paramétrer.
  Pour l'exemple l'utilisateur a choisi 3 étapes.

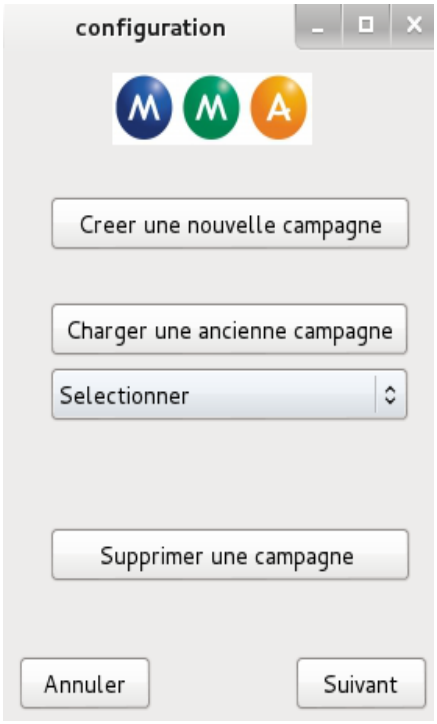


- Il ne vous reste plus qu'à paramétrer le nombre d'appel(s) pour le scénario choisi, les différentes étapes, en indiquant les DTMF et le temps de pause en chaque envoi de DTMF.

- Si vous voulez paramétrer un autre scénario, refaite la même manipulation pour le scénario 2 ,3, 4 et 5:

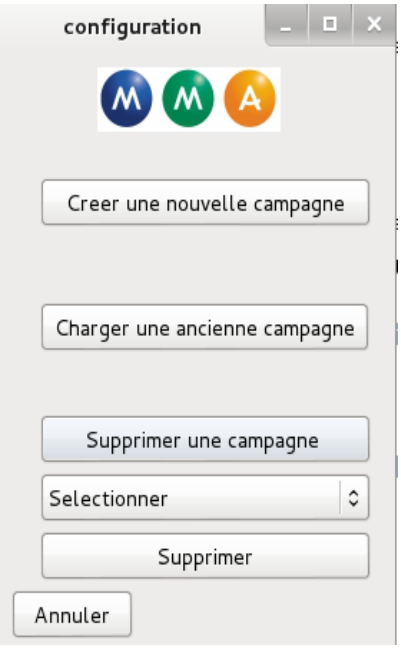
Une fois la configuration terminée, cliquez sur le bouton Appeler, vos appels seront passés.

15.2 Charger une ancienne campagne :



- Au lancement de l'application il vous suffit de cliquer sur charger une ancienne campagne, de la sélectionner et de cliquer sur suivant vous basculerez alors sur la fenêtre de la campagne avec les paramètres enregistrés précédemment.

15.3 Supprimer une campagne:



- Au lancement de l'application il vous suffit de cliquer sur Supprimer une campagne, de la sélectionner et de cliquer sur Supprimer, la campagne sera alors supprimée de la base de données.

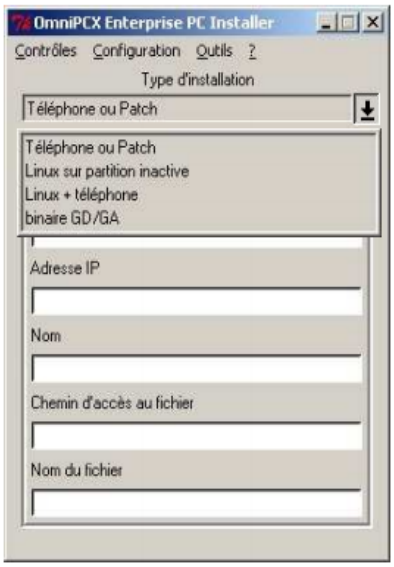
Glossaire

Thread: Les threads permettent l'exécution de plusieurs processus de manière simultané.

16. Annexe 1 - Installation

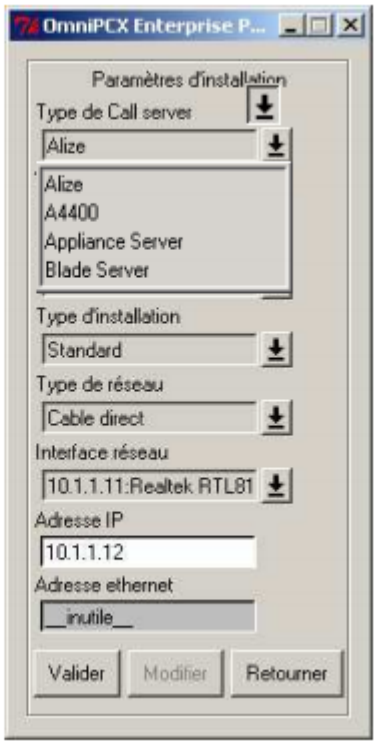
Il fallut charger tout le système sur la C.S. (Call Server), pour cella une application propriétaire à Alcatel-Lucent est mis à disposition des MMA. Pour y procéder, il est nécessaire de déterminer une IP fixe à la carte réseau qui permettra de faire la liaison entre le pc et la CS. Un second câble série DB9, nous sert à visualiser en détail les actions faites, en cours de traitement, via telnet ou HyperTerminal.

- Différents types de chargement
  - Téléphone ou Patch
    - Installation d'une version complète ou d'un patch
  - Linux sur partition inactive
    - Installation de Linux sur la deuxième partie du disque
  - Linux + Téléphone
    - Chargement complet automatisé
  - Binaire GD/GA
    - Rechargement du binaire d'une carte GD/GA



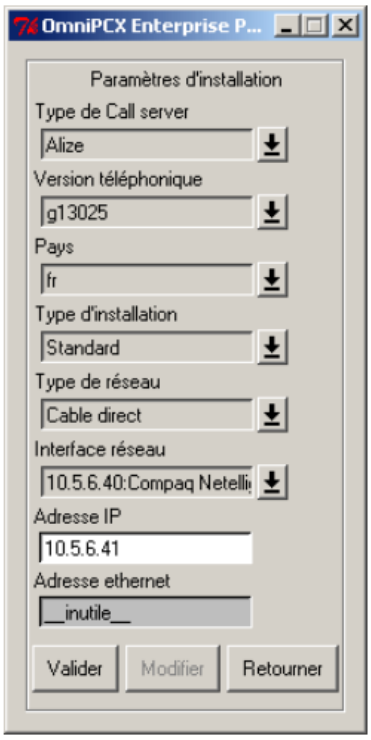
Dans notre cas le type de chargement choisi, est le mode Linux + téléphone, car il s'agit d'une nouvelle installation du système basé sur le noyau UNIX, ainsi que de la couche propriétaire du système de téléphonie.

- Différents types de Call Server
  - Alize (CPU-CS)
    - Requête bootp au démarrage
    - V24 facultative en chargement complet
  - A4400 (CPU 5/6/7)
    - Requête bootp au démarrage
    - V24 obligatoire en chargement complet
  - IP Appliance Serveur
    - Requête PXE au démarrage
    - V24 facultative en chargement complet
  - Blade Server
    - Requête PXE au démarrage
    - Console KVM obligatoire (pas de V24 disponible)



Puis dans un second temps, il faut préciser le type de CS dont nous disposons, dans notre cas d'une Alize, renseigner l'interface réseau ainsi que son adresse IP.

- Configuration de l'outil de chargement



← Choisir le type de CPU adapté (Alize/A4400/Appliance Server/Bade Server)

← Choisir la version téléphonique complète à installer **(Version + Patch statique + Patch dynamique)**

← Sélectionner le Pays

← Sélectionner Standard

← Sélectionner le type de connexion

← Modifier éventuellement la carte réseau à utiliser

← Modifier éventuellement l'adresse IP affectée à l'e-CS

← Indiquer l'adresse MAC de la CPU (mode Réseau client)

← Valider la configuration pour démarrer les serveurs

17. Annexe 2 – Procédure de suppression d'un utilisateur

1. Passer en mode super utilisateur : cmd -> su
2. cmd -> userdel [nom_de_l_utilisateur]

- Vérification de la suppression :

1. Liste des user (cmd -> more /etc/passwd)
2. Pour rechercher un user dans la liste :

   cmd -> fgrep [nom_de_l_utilisateur] /etc/passwd
3. Si aucun résultat ne ressort, l'utilisateur [nom_de_l_utilisateur] rechercher n'est pas/plus présent.